Citation APA:

Disponible à / Available at permalink : https://dipot.ulb.ac.be/dspace/bitstream/2013/210837/4/87f21324-db37-4aea-910d-58c3d88f885.txt

(English version below)

Cette thèse de doctorat a été numérisée par l’Université libre de Bruxelles. L’auteur qui s’opposerait à sa mise en ligne dans DI-fusion est invité à prendre contact avec l’Université (di-fusion@ulb.ac.be).

Dans le cas où une version électronique native de la thèse existe, l’Université ne peut garantir que la présente version numérisée soit identique à la version électronique native, ni qu’elle soit la version officielle définitive de la thèse.

DI-fusion, le Dépôt Institutionnel de l’Université libre de Bruxelles, recueille la production scientifique de l’Université, mise à disposition en libre accès autant que possible. Les œuvres accessibles dans DI-fusion sont protégées par la législation belge relative aux droits d’auteur et aux droits voisins. Toute personne peut, sans avoir à demander l’autorisation de l’auteur ou de l’ayant-droit, à des fins d’usage privé ou à des fins d’illustration de l’enseignement ou de recherche scientifique, dans la mesure justifiée par le but non lucratif poursuivi, lire, télécharger ou reproduire sur papier ou sur tout autre support, les articles ou des fragments d’autres œuvres, disponibles dans DI-fusion, pour autant que :
- Le nom des auteurs, le titre et la référence bibliographique complète soient cités;
- L’identifiant unique attribué aux métadonnées dans DI-fusion (permalink) soit indiqué;
- Le contenu ne soit pas modifié.

L’œuvre ne peut être stockée dans une autre base de données dans le but d’y donner accès ; l’identifiant unique (permalink) indiqué ci-dessus doit toujours être utilisé pour donner accès à l’œuvre. Toute autre utilisation non mentionnée ci-dessus nécessite l’autorisation de l’auteur de l’œuvre ou de l’ayant droit.

------------------------------------------------------ English Version -------------------------------------------------------------------

This Ph.D. thesis has been digitized by Université libre de Bruxelles. The author who would disagree on its online availability in DI-fusion is invited to contact the University (di-fusion@ulb.ac.be).

If a native electronic version of the thesis exists, the University can guarantee neither that the present digitized version is identical to the native electronic version, nor that it is the definitive official version of the thesis.

DI-fusion is the Institutional Repository of Université libre de Bruxelles; it collects the research output of the University, available on open access as much as possible. The works included in DI-fusion are protected by the Belgian legislation relating to authors’ rights and neighbouring rights. Any user may, without prior permission from the authors or copyright owners, for private usage or for educational or scientific research purposes, to the extent justified by the non-profit activity, read, download or reproduce on paper or on any other media, the articles or fragments of other works, available in DI-fusion, provided:
- The authors, title and full bibliographic details are credited in any copy;
- The unique identifier (permalink) for the original metadata page in DI-fusion is indicated;
- The content is not changed in any way.

It is not permitted to store the work in another database in order to provide access to it; the unique identifier (permalink) indicated above must always be used to provide access to the work. Any use not mentioned above requires the authors’ or copyright owners’ permission.

------------------------------------------------------ English Version -------------------------------------------------------------------
Designing Conventional, Spatial, and Temporal Data Warehouses: Concepts and Methodological Framework

Elzbieta Malinowski

Director: Prof. Esteban Zimányi
# Contents

Acknowledgements v

1 Introduction 1

1.1 Overview ................................................. 2

1.1.1 Conventional data warehouses ...................... 2

1.1.2 Spatial databases and spatial data warehouses ... 4

1.1.3 Temporal databases and temporal data warehouses 5

1.1.4 Conceptual modeling .................................. 6

1.1.5 Methodology for database and data warehouse design 6

1.2 Motivation and objectives of the thesis ............... 7

1.2.1 Motivation ............................................ 7

1.2.2 Objectives of the thesis ............................. 10

1.3 Scope and contributions of the thesis ................. 11

1.3.1 Conventional data warehouses ...................... 11

1.3.2 Spatial data warehouses ............................. 13

1.3.3 Temporal data warehouses ........................... 14

1.4 Thesis organization .................................... 15

2 Conventional data warehouses 17

2.1 Data warehouse structures and data manipulation .......... 18

2.1.1 Star, snowflake, and starflake schemas ............. 18

2.1.2 Aspects of data manipulation in DWs ................ 18

2.2 The MultiDimER model ................................ 20

2.2.1 Model definition .................................... 21

2.2.2 Model particularities ................................ 22

2.2.3 Metamodel of the MultiDimER model .............. 23

2.3 Mapping from conceptual to relational models .......... 23

2.3.1 Rationale for choosing relational model ............ 24

2.3.2 Mapping rules ...................................... 25

2.4 Data warehouse hierarchies ............................ 26

2.4.1 Simple hierarchies .................................. 27

2.4.2 Non-strict hierarchies .............................. 39

2.4.3 Multiple alternative hierarchies ................. 44

2.4.4 Parallel hierarchies ................................ 46

2.4.5 Hierarchy categorization ............................ 48

2.5 Example .................................................. 49
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6</td>
<td>Related work</td>
<td>51</td>
</tr>
<tr>
<td>2.7</td>
<td>Summary</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>Spatial data warehouses</td>
<td>57</td>
</tr>
<tr>
<td>3.1</td>
<td>Spatial databases: General concepts</td>
<td>58</td>
</tr>
<tr>
<td>3.2</td>
<td>Motivation</td>
<td>61</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Using the object-relational model</td>
<td>61</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Using spatial extensions of DBMSs</td>
<td>62</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Preserving semantics</td>
<td>63</td>
</tr>
<tr>
<td>3.3</td>
<td>Spatially-extended MultiDimER model</td>
<td>63</td>
</tr>
<tr>
<td>3.4</td>
<td>Spatial levels</td>
<td>65</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Conceptual representation</td>
<td>65</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Logical representation of spatial levels</td>
<td>66</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Mapping of levels with spatial descriptive attributes</td>
<td>68</td>
</tr>
<tr>
<td>3.5</td>
<td>Spatial hierarchies</td>
<td>68</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Conceptual representation</td>
<td>69</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Topological relationships between spatial levels</td>
<td>75</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Logical representation of spatial hierarchies</td>
<td>78</td>
</tr>
<tr>
<td>3.6</td>
<td>Spatial fact relationships and measures</td>
<td>83</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Conceptual representation</td>
<td>84</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Mapping of spatial fact relationships</td>
<td>90</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Mapping of spatial measures</td>
<td>91</td>
</tr>
<tr>
<td>3.7</td>
<td>Metamodel</td>
<td>93</td>
</tr>
<tr>
<td>3.8</td>
<td>Related work</td>
<td>95</td>
</tr>
<tr>
<td>3.9</td>
<td>Summary</td>
<td>97</td>
</tr>
<tr>
<td>4</td>
<td>Temporal data warehouses</td>
<td>99</td>
</tr>
<tr>
<td>4.1</td>
<td>Temporal databases: General concepts</td>
<td>100</td>
</tr>
<tr>
<td>4.2</td>
<td>Motivation</td>
<td>102</td>
</tr>
<tr>
<td>4.3</td>
<td>Temporal types in DWs</td>
<td>103</td>
</tr>
<tr>
<td>4.4</td>
<td>Temporal extension for the MultiDimER model</td>
<td>104</td>
</tr>
<tr>
<td>4.5</td>
<td>Mapping of temporal types</td>
<td>107</td>
</tr>
<tr>
<td>4.6</td>
<td>Temporal levels</td>
<td>109</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Conceptual representation</td>
<td>109</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Mapping of temporal levels</td>
<td>110</td>
</tr>
<tr>
<td>4.7</td>
<td>Temporal hierarchies</td>
<td>114</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Conceptual representation</td>
<td>114</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Mapping of child-parent relationships</td>
<td>119</td>
</tr>
<tr>
<td>4.8</td>
<td>Temporal measures</td>
<td>125</td>
</tr>
<tr>
<td>4.8.1</td>
<td>Temporal support for non-aggregated measures</td>
<td>125</td>
</tr>
<tr>
<td>4.8.2</td>
<td>Temporal support for aggregated measures</td>
<td>129</td>
</tr>
<tr>
<td>4.8.3</td>
<td>Mapping of fact relationships with temporal measures</td>
<td>132</td>
</tr>
<tr>
<td>4.9</td>
<td>Metamodel</td>
<td>134</td>
</tr>
<tr>
<td>4.10</td>
<td>Summary of mapping rules</td>
<td>136</td>
</tr>
<tr>
<td>4.11</td>
<td>Related work</td>
<td>137</td>
</tr>
<tr>
<td>4.12</td>
<td>Summary</td>
<td>140</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>A.4.3</td>
<td>Dimension definitions</td>
<td>213</td>
</tr>
<tr>
<td>A.4.4</td>
<td>Fact relationship definitions</td>
<td>214</td>
</tr>
<tr>
<td>A.4.5</td>
<td>Constraint definitions</td>
<td>214</td>
</tr>
<tr>
<td>A.5</td>
<td>Auxiliary functions</td>
<td>214</td>
</tr>
<tr>
<td>A.6</td>
<td>Semantics</td>
<td>214</td>
</tr>
<tr>
<td>A.6.1</td>
<td>Semantics of predefined data types</td>
<td>215</td>
</tr>
<tr>
<td>A.6.2</td>
<td>Semantic domains</td>
<td>215</td>
</tr>
<tr>
<td>A.6.3</td>
<td>Semantic functions</td>
<td>215</td>
</tr>
<tr>
<td>B</td>
<td>Formalization of the spatial MultiDimER model</td>
<td>220</td>
</tr>
<tr>
<td>B.1</td>
<td>Meta variables for the spatial extension</td>
<td>220</td>
</tr>
<tr>
<td>B.2</td>
<td>Abstract syntax</td>
<td>220</td>
</tr>
<tr>
<td>B.3</td>
<td>Examples using the abstract syntax</td>
<td>221</td>
</tr>
<tr>
<td>B.3.1</td>
<td>Level definitions</td>
<td>221</td>
</tr>
<tr>
<td>B.3.2</td>
<td>Child-parent relationship definitions</td>
<td>223</td>
</tr>
<tr>
<td>B.3.3</td>
<td>Dimension definitions</td>
<td>223</td>
</tr>
<tr>
<td>B.3.4</td>
<td>Fact relationship definitions</td>
<td>223</td>
</tr>
<tr>
<td>B.3.5</td>
<td>Constraint definitions</td>
<td>223</td>
</tr>
<tr>
<td>B.4</td>
<td>Semantics</td>
<td>224</td>
</tr>
<tr>
<td>B.4.1</td>
<td>The space model</td>
<td>224</td>
</tr>
<tr>
<td>B.4.2</td>
<td>Semantic domains</td>
<td>226</td>
</tr>
<tr>
<td>B.4.3</td>
<td>Semantic functions</td>
<td>226</td>
</tr>
<tr>
<td>C</td>
<td>Formalization of the temporal MultiDimER model</td>
<td>231</td>
</tr>
<tr>
<td>C.1</td>
<td>Meta variables for the temporal extension</td>
<td>231</td>
</tr>
<tr>
<td>C.2</td>
<td>Abstract syntax</td>
<td>231</td>
</tr>
<tr>
<td>C.3</td>
<td>Examples using the abstract syntax</td>
<td>232</td>
</tr>
<tr>
<td>C.3.1</td>
<td>Level definitions</td>
<td>232</td>
</tr>
<tr>
<td>C.3.2</td>
<td>Child-parent relationship definitions</td>
<td>234</td>
</tr>
<tr>
<td>C.3.3</td>
<td>Dimension definitions</td>
<td>235</td>
</tr>
<tr>
<td>C.3.4</td>
<td>Fact relationship definitions</td>
<td>235</td>
</tr>
<tr>
<td>C.3.5</td>
<td>Constraint definitions</td>
<td>235</td>
</tr>
<tr>
<td>C.4</td>
<td>Auxiliary functions</td>
<td>236</td>
</tr>
<tr>
<td>C.5</td>
<td>Semantics</td>
<td>236</td>
</tr>
<tr>
<td>C.5.1</td>
<td>The time model</td>
<td>236</td>
</tr>
<tr>
<td>C.5.2</td>
<td>Semantic domains</td>
<td>237</td>
</tr>
<tr>
<td>C.5.3</td>
<td>Semantic functions</td>
<td>237</td>
</tr>
<tr>
<td>D</td>
<td>Grapical notations</td>
<td>244</td>
</tr>
<tr>
<td>D.1</td>
<td>Notations for the ER model</td>
<td>244</td>
</tr>
<tr>
<td>D.2</td>
<td>Notations for relational and object-relational databases</td>
<td>246</td>
</tr>
<tr>
<td>D.3</td>
<td>Notations for conventional data warehouses</td>
<td>247</td>
</tr>
<tr>
<td>D.4</td>
<td>Notations for spatial data warehouses</td>
<td>249</td>
</tr>
<tr>
<td>D.5</td>
<td>Notations for temporal data warehouses</td>
<td>250</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>251</td>
</tr>
</tbody>
</table>
## 5 Methodology

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Database design</td>
<td>144</td>
</tr>
<tr>
<td>5.1.1 Conventional database design</td>
<td>144</td>
</tr>
<tr>
<td>5.1.2 Spatial and temporal database design</td>
<td>145</td>
</tr>
<tr>
<td>5.2 Current approaches for data warehouse design</td>
<td>146</td>
</tr>
<tr>
<td>5.2.1 Data warehouse lifecycle</td>
<td>146</td>
</tr>
<tr>
<td>5.2.2 Data mart and data warehouse design</td>
<td>148</td>
</tr>
<tr>
<td>5.2.3 Design phases</td>
<td>149</td>
</tr>
<tr>
<td>5.2.4 Methods for requirements specifications and DW design</td>
<td>150</td>
</tr>
<tr>
<td>5.3 A methodology for conventional data warehouses</td>
<td>151</td>
</tr>
<tr>
<td>5.4 A motivating case study</td>
<td>152</td>
</tr>
<tr>
<td>5.5 Requirements specification and conceptual design</td>
<td>154</td>
</tr>
<tr>
<td>5.5.1 Demand-driven approach</td>
<td>154</td>
</tr>
<tr>
<td>5.5.2 Supply-driven approach</td>
<td>166</td>
</tr>
<tr>
<td>5.5.3 Demand/supply-driven approach</td>
<td>173</td>
</tr>
<tr>
<td>5.6 Logical design</td>
<td>176</td>
</tr>
<tr>
<td>5.6.1 Logical representation of DW schemas</td>
<td>176</td>
</tr>
<tr>
<td>5.6.2 Defining ETL processes</td>
<td>180</td>
</tr>
<tr>
<td>5.7 Physical design</td>
<td>181</td>
</tr>
<tr>
<td>5.7.1 DW schema implementation</td>
<td>181</td>
</tr>
<tr>
<td>5.7.2 ETL processes and staging area implementation</td>
<td>186</td>
</tr>
<tr>
<td>5.8 A methodology for spatial and temporal DWs</td>
<td>187</td>
</tr>
<tr>
<td>5.8.1 Spatial extension</td>
<td>187</td>
</tr>
<tr>
<td>5.8.2 Temporal extension</td>
<td>191</td>
</tr>
<tr>
<td>5.9 Related work</td>
<td>194</td>
</tr>
<tr>
<td>5.9.1 Overall process</td>
<td>194</td>
</tr>
<tr>
<td>5.9.2 Requirements specifications</td>
<td>196</td>
</tr>
<tr>
<td>5.9.3 Conceptual, logical, and physical level design</td>
<td>199</td>
</tr>
<tr>
<td>5.9.4 Spatial and temporal design methodologies</td>
<td>200</td>
</tr>
<tr>
<td>5.10 Summary</td>
<td>201</td>
</tr>
</tbody>
</table>

## 6 Conclusions and future work

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Conclusions</td>
<td>203</td>
</tr>
<tr>
<td>6.2 Future work</td>
<td>206</td>
</tr>
<tr>
<td>6.2.1 Conventional data warehouses</td>
<td>206</td>
</tr>
<tr>
<td>6.2.2 Spatial data warehouses</td>
<td>207</td>
</tr>
<tr>
<td>6.2.3 Temporal and spatio-temporal data warehouses</td>
<td>208</td>
</tr>
<tr>
<td>6.2.4 Methodology</td>
<td>209</td>
</tr>
</tbody>
</table>

## A Formalization of the MultiDimER model

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Notations and assumptions</td>
<td>210</td>
</tr>
<tr>
<td>A.2 Meta variables</td>
<td>210</td>
</tr>
<tr>
<td>A.3 Abstract syntax</td>
<td>211</td>
</tr>
<tr>
<td>A.4 Examples using the abstract syntax</td>
<td>213</td>
</tr>
<tr>
<td>A.4.1 Level definitions</td>
<td>213</td>
</tr>
<tr>
<td>A.4.2 Child-parent relationship definitions</td>
<td>213</td>
</tr>
</tbody>
</table>
Acknowledgements

First of all, I would like to express my gratitude to the Universe for the opportunity and challenging events that allow me to grow both professionally and personally.

I am also very thankful for my family's support. This work would be impossible to finish without the constant, daily encouragement from my husband. Thanks, Yamil, for believing in me more than I could believe in myself. I am also grateful to you for helping me balance my life between different everyday activities. Thanks, Bartosz, for being such a self-sufficient son during these four years. You are a strong guy who knows how to face challenges with mom constantly making you move to a different country, learn new languages, and shape your new life. Thanks, Carito, my unbelievable daughter, for taking all the responsibilities in Costa Rica. Without your help I would not have been able to study in Belgium.

I would also like to express my gratitude to the Director of this thesis, Esteban Zimányi. His critical observations helped me improve the analytical skills required for the research process as well as the writing skills required for disseminating the results. I also appreciate the freedom that Esteban gave me in choosing research directions and methods. I believe that the ability that I acquired during my doctoral studies of looking for new problems and finding their solutions is an important skill to achieve future professional goals. I also had a privilege to have Esteban's friendship for four years and to witness the birth of his first daughter, Elenita. I will take with me the memory of these moments of hard work and fun.

Special thanks goes to my research committee, Dr. Jef Wijsen and Dr. Roel Wuyts. You cannot imagine how much your encouragement and appreciation of my academic effort meant to me during the presentation sessions over these four years. I am also particularly grateful for the valuable comments from Professor Wijsen who made me analyze my proposal more in-depth to improve its quality. I have also received important advice from professor Wuyts who contributed to the accuracy and readiness of this thesis. Thanks also both of you for unbelievable support in different occasions during the last period of writing my dissertation.

I would also like to show my appreciation to Dr. Marco Dorigo and Professor Jean-Luc Hainaut for the help in the final phase of writing my dissertation. The detailed revision of professor Dorigo and his positive feedback motivated me to keep on working to improve my thesis. I would like to acknowledge that, in spite of time constraints and your busy schedule, you were able to read this thesis and give me your valuable comments. Special thanks goes also to Professor Hainaut. I was fortunate to have you in my committee and to benefit from your expertise. Thank you for dedicating your valuable time to revise my work and to give me additional recommendations for increasing its quality.
I would also like to express my gratitude to many anonymous reviewers who evaluated my articles, either accepting or rejecting them, for conferences, workshops, and journals. The feedback I received from them helped to improve the quality of this thesis and to expand my research scope. Thanks you for this enormous effort, which I know was time-consuming and did not have a direct appreciation gesture.

I am also thankful for the continuous administrative support from Natasha Vander Heyden. I know very few people who help the way you do without asking recognition for your job.

Last, but not least, I am dedicating my appreciation to two universities: the Université Libre de Bruxelles, specially to Service de Coopération, and the Universidad de Costa Rica. Thanks to the former I received the scholarship that allowed me to accomplish the dream of my professional life: a PhD degree. This financial and human support gave me the necessary comfort to focus on my research. I would like to express my gratitude to Mme. Dominique Mertens and Larondelle Arlette from the Service de Coopération for helping me with all administrative issues. Thanks also to Dominique for being such a special person and sharing her friendship with me. I strongly believe that the time and effort you invested in me and financial support you gave me will replicate enormously when I share my knowledge with my students at the Universidad de Costa Rica.

Special thanks to the Universidad de Costa Rica that provided financial support to my family. I believe this university is one of a kind, as it strongly supports the human side of scientists and encourages them to develop their professional growth by simultaneously promoting family union. I would like to particularly express my appreciation to the former and current directors of the Oficina de Asuntos Internacionales (International Affairs Bureau) of the Universidad de Costa Rica, Dr. Manuel Murillo y Dra Ana Sittenfeld, respectively. Thank you for providing very efficient and rapid administrative procedures and for offering all the necessary conditions to complete my doctoral studies. Special thanks goes also to Patricia Alfaro, Fátima Acosta, and Yamilet Damazio, from the same bureau. You were very supportive and patient with all the questions I had during these four years of study.
Abstract

Decision support systems are interactive, computer-based information systems that provide data and analysis tools in order to better assist managers on different levels of organization in the process of decision making. Data warehouses (DWs) have been developed and deployed as an integral part of decision support systems.

A data warehouse is a database that allows to store high volume of historical data required for analytical purposes. This data is extracted from operational databases, transformed into a coherent whole, and loaded into a DW during the extraction-transformation-loading (ETL) process.

DW data can be dynamically manipulated using on-line analytical processing (OLAP) systems. DW and OLAP systems rely on a multidimensional model that includes measures, dimensions, and hierarchies. Measures are usually numeric additive values that are used for quantitative evaluation of different aspects about organization. Dimensions provide different analysis perspectives while hierarchies allow to analyze measures on different levels of detail.

Nevertheless, currently, designers as well as users find difficult to specify multidimensional elements required for analysis. One reason for that is the lack of conceptual models for DW and OLAP system design, which would allow to express data requirements on an abstract level without considering implementation details. Another problem is that many kinds of complex hierarchies arising in real-world situations are not addressed by current DW and OLAP systems.

In order to help designers to build conceptual models for decision-support systems and to help users in better understanding the data to be analyzed, in this thesis we propose the MultiDimER model – a conceptual model used for representing multidimensional data for DW and OLAP applications. Our model is mainly based on the existing ER constructs, for example, entity types, attributes, relationship types with their usual semantics, allowing to represent the common concepts of dimensions, hierarchies, and measures. It also includes a conceptual classification of different kinds of hierarchies existing in real-world situations and proposes graphical notations for them.

On the other hand, currently users of DW and OLAP systems demand also the inclusion of spatial data, visualization of which allows to reveal patterns that are difficult to discover otherwise. The advantage of using spatial data in the analysis process is widely recognized since it allows to reveal patterns that are difficult to discover otherwise.

However, although DWs typically include a spatial or a location dimension, this dimension is usually represented in an alphanumeric format. Furthermore, there is still a lack of a systematic study that analyze the inclusion as well as the management of hierarchies and measures that are represented using spatial data.
With the aim of satisfying the growing requirements of decision-making users, we extend the MultiDimER model by allowing to include spatial data in the different elements composing the multidimensional model. The novelty of our contribution lays in the fact that a multidimensional model is seldom used for representing spatial data. To succeed with our proposal, we applied the research achievements in the field of spatial databases to the specific features of a multidimensional model. The spatial extension of a multidimensional model raises several issues, to which we refer in this thesis, such as the influence of different topological relationships between spatial objects forming a hierarchy on the procedures required for measure aggregations, aggregations of spatial measures, the inclusion of spatial measures without the presence of spatial dimensions, among others.

Moreover, one of the important characteristics of multidimensional models is the presence of a time dimension for keeping track of changes in measures. However, this dimension cannot be used to model changes in other dimensions. Therefore, usual multidimensional models are not symmetric in the way of representing changes for measures and dimensions. Further, there is still a lack of analysis indicating which concepts already developed for providing temporal support in conventional databases can be applied and be useful for different elements composing a multidimensional model.

In order to handle in a similar manner temporal changes to all elements of a multidimensional model, we introduce a temporal extension for the MultiDimER model. This extension is based on the research in the area of temporal databases, which have been successfully used for modeling time-varying information for several decades. We propose the inclusion of different temporal types, such as valid and transaction time, which are obtained from source systems, in addition to the DW loading time generated in DWs. We use this temporal support for a conceptual representation of time-varying dimensions, hierarchies, and measures. We also refer to specific constraints that should be imposed on time-varying hierarchies and to the problem of handling multiple time granularities between source systems and DWs.

Furthermore, the design of DWs is not an easy task. It requires to consider all phases from the requirements specification to the final implementation including the ETL process. It should also take into account that the inclusion of different data items in a DW depends on both, users' needs and data availability in source systems. However, currently, designers must rely on their experience due to the lack of a methodological framework that considers above-mentioned aspects.

In order to assist developers during the DW design process, we propose a methodology for the design of conventional, spatial, and temporal DWs. We refer to different phases, such as requirements specification, conceptual, logical, and physical modeling. We include three different methods for requirements specification depending on whether users, operational data sources, or both are the driving force in the process of requirement gathering. We show how each method leads to the creation of a conceptual multidimensional model. We also present logical and physical design phases that refer to DW structures and the ETL process.

To ensure the correctness of the proposed conceptual models, i.e., with conventional data, with the spatial data, and with time-varying data, we formally define them providing their syntax and semantics. With the aim of assessing the usability of our conceptual model including representation of different kinds of hierarchies as well
as spatial and temporal support, we present real-world examples. Pursuing the goal that the proposed conceptual solutions can be implemented, we include their logical representations using relational and object-relational databases. Further, to show the applicability of the proposed methodology and different approaches used for requirements specification, we include several real-world examples related to universities.
List of Figures

2.1 An example of a starflake schema......................................................... 19
2.2 Notations for multidimensional model: a) level, b) hierarchy, c) analysis criterion, d) cardinalities, and e) fact relationship with associated measures.......................................................................................................... 21
2.3 A conceptual multidimensional schema of a Sales DW from Figure 2.1. 22
2.4 Metamodel of the MultiDimER model.................................................. 24
2.5 A symmetric hierarchy: a) schema and b) example of instances......... 28
2.6 An asymmetric hierarchy: a) schema and b) examples of instances.... 30
2.7 Transformation of the asymmetric hierarchy from Figure 2.6 b) to a symmetric one using placeholders................................................................. 31
2.8 Parent-child representation of an asymmetric hierarchy: a) relational schema b) MultiDimER representation, and c) instances from Figure 2.6 b). ........................................................................................................ 32
2.9 ER representation of a dimension with different kinds of customers. . 33
2.10 A generalized hierarchy for a sales company a) schema and b) examples of instances................................................................. 34
2.11 An example of the generalized hierarchy without a joining level. ...... 35
2.12 A non-covering hierarchy a) schema and b) examples of instances. ... 36
2.13 Relations for the generalized hierarchy from Figure 2.10. ................... 37
2.14 Improved relational representation for the generalized hierarchy of Figure 2.10. ........................................................................................ 38
2.15 A symmetric non-strict hierarchy: a) schema and b) example of instances. 39
2.16 Aggregations of a sales measure for a) strict and b) non-strict hierarchies. 40
2.17 A symmetric non-strict hierarchy with a distributing factor. ............. 41
2.18 Relational tables with a bridge table for the symmetric non-strict hierarchy of Figure 2.17. ........................................................................... 42
2.19 Transformation of a non-strict hierarchy into a strict hierarchy, a) relational tables and b) the MultiDimER representation. ................. 43
2.20 Double counting problem for non-strict hierarchies. ......................... 43
2.21 A multiple alternative hierarchy: a) conceptual schema and b) relations. 45
2.22 Parallel independent hierarchy composed of a symmetric and a non-covering hierarchies. ................................................................. 46
2.23 A conceptual schema of parallel dependent hierarchies composed of a symmetric and a non-covering hierarchies......................... 47
2.24 A conceptual schema of parallel dependent hierarchies composed of a symmetric and a non-covering hierarchies. ...................... 48
<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.25</td>
<td>Hierarchy classification</td>
<td>49</td>
</tr>
<tr>
<td>2.26</td>
<td>Example of schema containing several hierarchies reusing hierarchy levels</td>
<td>50</td>
</tr>
<tr>
<td>3.1</td>
<td>Pictograms for: a) spatial data types and b) topological relationships</td>
<td>59</td>
</tr>
<tr>
<td>3.2</td>
<td>An example of a multidimensional schema with spatial elements</td>
<td>64</td>
</tr>
<tr>
<td>3.3</td>
<td>Examples of a) a spatial level, b) a spatial level with a spatial attribute, and c) a thematic level with a spatial attribute</td>
<td>65</td>
</tr>
<tr>
<td>3.4</td>
<td>A spatial level: a) the OR representation of members and b) a member with an island</td>
<td>66</td>
</tr>
<tr>
<td>3.5</td>
<td>A symmetric spatial hierarchy: a) schema and b) examples of instances</td>
<td>69</td>
</tr>
<tr>
<td>3.6</td>
<td>An asymmetric spatial hierarchy: a) schema and b) examples of instances</td>
<td>70</td>
</tr>
<tr>
<td>3.7</td>
<td>A generalized spatial hierarchy: a) schema and b) examples of instances</td>
<td>71</td>
</tr>
<tr>
<td>3.8</td>
<td>A symmetric non-strict spatial hierarchy: a) schema and b) example of instances</td>
<td>72</td>
</tr>
<tr>
<td>3.9</td>
<td>A multiple alternative spatial hierarchy formed by two non-strict symmetric hierarchies</td>
<td>73</td>
</tr>
<tr>
<td>3.10</td>
<td>Parallel independent spatial hierarchies associated to one dimension</td>
<td>73</td>
</tr>
<tr>
<td>3.11</td>
<td>Parallel dependent spatial hierarchies in the Client dimension</td>
<td>74</td>
</tr>
<tr>
<td>3.12</td>
<td>A spatial hierarchy in the Client dimension</td>
<td>75</td>
</tr>
<tr>
<td>3.13</td>
<td>Dimensions with hierarchies sharing spatial levels</td>
<td>76</td>
</tr>
<tr>
<td>3.14</td>
<td>Classification of topological relationships for aggregation procedures</td>
<td>77</td>
</tr>
<tr>
<td>3.15</td>
<td>Example of the relationship between non-spatial and spatial levels</td>
<td>78</td>
</tr>
<tr>
<td>3.16</td>
<td>Example of relationship between two spatial levels</td>
<td>80</td>
</tr>
<tr>
<td>3.17</td>
<td>Example of schema for analysis of the city transportation service</td>
<td>84</td>
</tr>
<tr>
<td>3.18</td>
<td>A multidimensional conceptual schema with a spatial measure: Location</td>
<td>85</td>
</tr>
<tr>
<td>3.19</td>
<td>Spatial dimension: another variant of a schema for analysis of accidents</td>
<td>87</td>
</tr>
<tr>
<td>3.20</td>
<td>A multidimensional schema for analysing closeness of cities and highways</td>
<td>90</td>
</tr>
<tr>
<td>3.21</td>
<td>Metamodel of the spatially-extended MultiDimER model</td>
<td>93</td>
</tr>
<tr>
<td>4.1</td>
<td>Examples of an event and a state</td>
<td>101</td>
</tr>
<tr>
<td>4.2</td>
<td>An example of conceptual schema for a TDW</td>
<td>107</td>
</tr>
<tr>
<td>4.3</td>
<td>Representation in the MultiDimER model of a) temporal attributes and b) level lifespan</td>
<td>110</td>
</tr>
<tr>
<td>4.4</td>
<td>A level with temporal attributes: a) and c) the ER schemas, and b) and d) the object-relational representations</td>
<td>111</td>
</tr>
<tr>
<td>4.5</td>
<td>A level with lifespan: a) the ER schema and b) the object-relational representation</td>
<td>114</td>
</tr>
<tr>
<td>4.6</td>
<td>An example of a non-temporal relationship between temporal levels</td>
<td>115</td>
</tr>
<tr>
<td>4.7</td>
<td>An example of incorrect analysis scenario: a) before and b) after splitting categories</td>
<td>115</td>
</tr>
<tr>
<td>4.8</td>
<td>An example of the MultiDimER schema of temporal relationships between non-temporal levels</td>
<td>116</td>
</tr>
<tr>
<td>4.9</td>
<td>An example of dangling references: a) before and b) after splitting a section</td>
<td>116</td>
</tr>
<tr>
<td>4.10</td>
<td>Examples of different temporal relationships between non-temporal levels: a) evolution and b) extension</td>
<td>116</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

4.11 An example of temporal relationships between temporal levels. .......... 117
4.12 Snapshot and lifespan cardinalities between hierarchy levels. ................. 119
4.13 A hierarchy with a non-temporal relationship: a) the ER schema and b) the object-relational representation. .......................................................... 120
4.14 Temporal relationships linking non-temporal levels: a) the MultiDimER schema, b) the ER schema, and c) alternative object-relational representations. .......................................................... 122
4.15 Temporal relationships linking non-temporal levels: a) and b) different versions of the MultiDimER schema and c) the object-relational representation for the Employee level. .......................................................... 123
4.16 Temporal relationship between temporal levels: a) the MultiDimER model and b) the object-relational representation. .......................................................... 124
4.17 Inclusion of DWLT for measures. .............................................................. 126
4.18 Inclusion of VT for measures representing a) event and b) state models. 127
4.19 An example of the usefulness of having VT and DWLT. ........................... 128
4.20 An example of a temporal DW for insurance company. ......................... 128
4.21 An example of the usefulness of having VT, TT, and DWLT for temporal DWs. .......................................................... 129
4.22 Example of coercion function for salary. .................................................. 131
4.23 Temporal measures: a) the ER representation, b) the relational table for the Sales measure, and c) the object-relational representation. .......................................................... 133
4.24 Metamodel of the temporally-extended MultiDimER model. ..................... 134
5.1 DW lifecycle diagram. .......................................................... 146
5.2 Data warehouse design phases. .................................................. 151
5.3 General phases in the demand-driven approach for a) requirements specification and b) conceptual design for conventional DWs. ............................... 155
5.4 Initial multidimensional schema for the analysis of international publications. .......................................................... 160
5.5 Initial multidimensional schema for the analysis of international conferences. .......................................................... 161
5.6 Initial multidimensional schema for the analysis of international projects. 162
5.7 An example of a data mart schema. .......................................................... 163
5.8 General phases in the supply-driven approach for a) requirements specification and b) conceptual design of conventional DWs. .......................................................... 166
5.9 Part of the ER schema of the underlying operational DB. ......................... 169
5.10 The derived DW schema from the ER schema of Figure 5.9. ...................... 170
5.11 General phases in the demand/supply-driven approach for conventional DWs. .......................................................... 174
5.12 General phases in the demand-driven approach for spatial DWs. ............ 188
5.13 General phases in the supply-driven approach for spatial DWs. ............. 189
5.14 General phases in the demand/supply-driven approach for spatial DWs. 191
A.1 Example of schema containing several hierarchies reusing hierarchy levels. 212
B.1 Example of a model including spatial and non-spatial elements. ............... 222
B.2 Example of a realm. .......................................................... 225
LIST OF FIGURES

C.1 Example of a model including temporal and non-temporal elements. . . 233
List of Tables

2.1 Comparison of implementations of non-strict hierarchies using a bridge table and separate tables. ............................................. 44
2.2 Comparison of the conceptual models including different kinds of hierarchies. ................................................................. 52
3.1 Examples of queries for conventional DWs and spatial DWs. ........... 86
3.2 Differences between the schemas of Figure 3.18 and Figure 3.19. ....... 88
4.1 Temporal types in source systems that may be used in temporal DWs. 105
4.2 Temporal types allowed in the MultiDimER model. ...................... 135
5.1 Specification of conformed levels. ............................................... 162
Chapter 1

Introduction

Organizations are becoming increasingly complex in their management and solving problems for the purpose of attaining established goals. This situation compels the users to utilize analysis systems and tools that better support their decisions. The decision making systems (DSSs) aim to help managers at different levels of organization in the analysis of information. The basic idea behind this kind of systems is to collect vast amount of data and reduce it to a form that could be used to analyze the behavior of the organization [38].

Since the early-1990s data warehouses (DWs) have been developed and deployed as an integral part of modern DSSs [140, 187]. A DW provides an infrastructure that enables users to obtain efficient and accurate responses to complex queries. Different systems and tools can be used for accessing and analyzing the data contained in DWs. For example, on-line analytical processing (OLAP) systems allow to interactively query and automatically aggregated the DW data. In this way, decision-making users can easily access the required information and analyze it on different levels of detail.

Nevertheless, even though well-organized and aggregated data can facilitate analysis process, there are two more aspects that are important for decision-making users and should be considered in DWs. Firstly, in many situations representing data spatially, e.g., in a map, could help to reveal patterns that are difficult to discover otherwise. For example, representing in a map customers' locations may help to find that most customers not necessarily buy products in the closest stores. Secondly, there is a need to store all historical data without overwriting the old values. For example, storing all changes to products' ingredients may help to analyze whether these changes have some influence on sales.

Further, DW systems are very complex systems and their development is a difficult task. DW developer teams must know not only what data and what kind of analysis users require for supporting the decision making process but also which phases should be followed to ensure success in the DW development. Therefore, it is necessary to provide a model that facilitates the communication between users and DW designers, and a methodology for DW development.

In this chapter, we refer to aspects that motivate us to develop a conceptual multi-dimensional model with spatial and temporal extensions and to propose a methodology for their design. First, in Section 1.1 we briefly describe general concepts related to data warehouses and to their spatial and temporal extensions. We also refer to con-
ceptual modeling and the methodology used for designing conventional, spatial, and temporal data warehouses. Then, in Section 1.2 we present the motivations and objectives of this thesis. Finally, in Section 1.3 we describe the scope and contributions of this thesis and in Section 1.4 we refer to its organization.

1.1 Overview

1.1.1 Conventional data warehouses

The importance of data analysis has grown significantly in recent years as business in all sectors has discovered the competitive advantage that just-in-time information can give for the decision-making process. However, traditional database (DB) systems, called operational or transactional, do not satisfy the requirements for data analysis. They support daily business operations and the primary concern of such databases is to ensure concurrent access and recovery techniques that guarantee data consistency. Typical operational databases contain detailed data, do not include historical data, and since they are usually highly normalized, they perform poorly when executing complex queries that need to join many relational tables or to aggregate large volumes of data. Further, when users require to analyze the behavior of an organization as a whole, the data from different operational systems must be joined. This can be a difficult task to accomplish because of differences in data definition and content.

Data warehouses were proposed in order to better respond to the growing demands of decision-making users. A data warehouse (DW) is a collection of subject-oriented, integrated, non-volatile, and time-variant data to support management's decisions [86].

- **Subject orientation** means that the development of DWs is done according to the analytical necessities of managers at different levels of the decision-making process, i.e., it is oriented by a subject of analysis. Subjects can vary depending on the kind of business activities, for example, analysis of sales in different kinds of businesses, analysis of clients' behaviors in using banking services, analysis of buying patterns in retail businesses, analysis of a railroad system utilization in transportation companies, etc. Within an organization several subjects of analysis may be included in the same DW. For example, a DW in a retail company may have data related to analysis of products' purchase, inventory, and sales. Modeling DW data according to subjects of analysis differs from conventional modeling used for operational databases where focus is on specific functions that applications should handle, for example, to deposit or to withdraw money, to register product sales, etc.

- **Integration** represents the complex effort to join data from different operational and external systems and to solve the problems of the definition and content of data, such as the differences in data format, data codification, synonyms (fields with different names but the same data), homonyms (fields with the same name but different meaning), multiplicity of data occurrences, and many others. In operational databases these problems usually do not exist since they are typically solved in the database design phase.
1.1. OVERVIEW

- **Non-volatility** ensures data durability by disallowing data modification and removal, thus expanding data utility for a longer period of time than operational systems can usually offer. In the latter, data is often kept only for a short period of time, for example, from 2 to 6 months as it is required for the daily business, and it may be overwritten when necessary.

- **Time-variation** indicates the possibility to keep different values of the same information and the time when changes to these values have occurred. For example, a DW in a bank might store the information about the average balance of a client’s account during different months for the period of several years. At the contrary, operational databases may not have explicit temporal support since it may be difficult to implement or it is not necessary. For example, it might be unnecessary to store information about previous employee’s salary.

Operational databases have typically used traditional database techniques for their design, i.e., entity-relationship (ER) modeling and normalization. However, several authors (e.g., [101, 197]) indicate that these paradigms are not well suited for DW applications. DWs should be modeled in a way that ensures a better understanding of data for analysis purposes and gives better performance for complex queries needed for analyzing this data. In order to meet these expectations, a **multidimensional model** was proposed. This model is usually represented as relational tables with a specialized structure called a *star schema*.

A star schema consists of the **fact table**, which links to several other relations called **dimension tables**. The fact table represents the focus of analysis (for example, analysis of sales in stores) and typically includes attributes called **measures**. Measures are usually numeric values that allow to perform quantitative evaluation of different aspects in organizations. For example, measures of sales or quantity might help to analyze selling activities in different stores.

Dimension tables contain attributes that allow to see the measures from different perspectives. For example, a time dimension can be used for analyzing changes in sales during different periods of time while a location dimension can be used to analyze sales according to geographic distribution of stores. A user may combine different analysis perspectives (i.e., dimensions) according to his needs. For example, he may require the information about sales of Milk (the product dimension) in July (the time dimension) in all store locations (the store dimension).

On the other hand, since the decision-making users usually start from a general view of data and then, if required, the detailed explorations follow, dimensions may include attributes that form hierarchies, such as month-year in the time dimension or city-state in the location dimension. While traversing a hierarchy, an aggregation of measures takes place. For example, moving in a hierarchy from a month to a year will give aggregated values of sales for the month and for the year, respectively.

On-line analytical processing (OLAP) systems provide flexible interactions with end-users and dynamic manipulations of the data contained in DWs. They facilitate complex query formulation that may involve a very large amount of data. This data is examined and aggregated in order to find patterns or trends of importance to the organization.
CHAPTER 1. INTRODUCTION

OLAP systems have typically been implemented using two technologies: relational OLAP (ROLAP) and multidimensional OLAP (MOLAP). The former stores data in a relational database management system (DBMS) while the latter uses vendor-specific data structures. However, in order to exploit ROLAP as well as MOLAP systems to their full capabilities, data should be represented using a multidimensional model, i.e., representing facts with measures, dimensions, and hierarchies. In particular, the specification of hierarchies is important since in this way OLAP systems are able to perform automatic aggregations of measures while traversing hierarchies.

1.1.2 Spatial databases and spatial data warehouses

Over the years, spatial data has increasingly become part of operational and analytical systems in different areas, such as public administration, transportation networks, military applications, environmental systems, public health, among others. This kind of data can represent, on the one hand, geographic objects, i.e., objects located on Earth's surface, such as mountains, cities, rivers, or geographic phenomena, such as temperature, precipitation, altitude. On the other hand, it can stand for different kinds of objects, such as parts of human body, parts of a house, parts of an engine, among others. Currently, due to technological advances, the amount of available spatial data is growing considerably, for example, satellite images, medical images, location data from remote sensing systems, such as Global Positioning Systems (GPS).

The management of spatial data is usually carried out by spatial databases (spatial DBs) or geographic information systems (GISs). Since the latter are used for storing and manipulating geographic objects and phenomena, in the following we will use the term spatial DBs as a more general term than GISs.

Spatial DBs include a component that allows to store spatial data describing their location, shape, and size. Further, these systems have a set of functions and operators that allow users to query spatial data. Queries can be simple referring to spatial characteristics of individual objects, such as area, or can be more complex requiring operations on two or more spatial objects, such as spatial join. Topological relationships between spatial objects are also important in spatial DB applications. Topological relationships, such as intersection, inside, meet, are those relationships that do not change when spatial objects are rotated, scaled, etc. For example, two roads can intersect, a lake can be inside of a state, two countries can meet when they have common borders.

However, even though spatial DBs can offer sophisticated management of spatial data, these systems are usually used in daily business queries and operations similarly to conventional operational databases. Spatial DBs are not well suitable for supporting the decision-making process. As a consequence a new field, called spatial data warehouses (spatial DWs), has emerged combining DW and spatial DB technologies. Spatial DWs allow to exploit the capabilities of both systems for improving data analysis, visualization, and manipulation. For example, DWs provide efficient access methods and management of high volumes of data. On the other hand, spatial DBs have a long experience in managing spatial data, and there is extensive research referring to spatial index structures, storage management, and dynamic query formulation.
1.1. OVERVIEW

1.1.3 Temporal databases and temporal data warehouses

Many applications require the storage, manipulation, and retrieval of information that varies over time. These applications should not only be able to represent changes to data but also the time when they have occurred. For example, land management systems need to represent land distributions and the time when changes have occurred, i.e., how and when the land have been split, merged or when an owner has been changed; health systems may include historical information about patients’ clinical records including different medical exams and the time when they were taken; reservation systems must manage time in order to offer different services for their clients; and so on.

Since current DBMSs do not provide special facilities for storing, managing, and updating time-varying data, the representation of these changes and the time when they have occurred is left to programming skills of implementers. Therefore, complex structures and significant programming effort are required for the correct management of data varying over time.

On the other hand, temporal DBs allow to represent and manage time-varying information. These kinds of databases include temporal support by means of two temporal types. The first type, called valid time, indicates the time when data was (or is or will be) valid in the modeled reality, for example, the time when a student took some specific course. Another temporal type, called transaction time, specifies the time when a data is current in a database, for example, when the information that the student is taking the course is stored in the databases.

Further, one of the characteristics of temporal types is the time precision or granularity. For example, an employee’s salary can correspond to month May 2006, thus the granularity used for this valid time is month. In a similar way, transaction time has defined its precision that is not necessarily the same as the valid time granularity since transaction time is system defined. For example, the salary corresponding to May 2006 may be stored in the database using the millisecond granularity.

Different approaches exist for temporal support in DBs depending on whether the modifications to the relational model are required or not. On the one hand, temporal support can be added by means of what might be thought of hidden attributes, i.e., they are hidden in the sense that they cannot be referenced by simple names in the usual way [189]. Using this approach, temporal data is handled differently than data usually included in relational DBs. Therefore, in order to consider temporal semantics as an integral part of databases, special DBMSs that include the different temporal types, adequate storage, indexing, query languages, etc. are required [189].

On the other hand, another approach does not involve any changes to the classical relational model [39] and treats temporal data just like data of any other kind, i.e., temporal support is implemented by means of explicit attributes. This approach introduces a new interval data type and provides a set of new operators and extensions of existing ones for managing time-varying data.

Temporal DBs allow to manage historical data; however, they do not offer facilities for supporting the decision-making process when aggregations of high volumes of

\[^1\) They are usually called time dimensions; however, we use the term “dimension” in the multidimensional context.
historical data are required. Therefore, bringing together the research achievements of temporal DBs and DWs leads to a new field called temporal DWs. In this way, temporal semantics forms an integral part of temporal DWs in a similar way as it is the case for temporal DBs.

1.1.4 Conceptual modeling

The conventional database design process includes creation of database schemas at three different levels: conceptual, logical, and physical [48]. The conceptual schema is a concise description of the data requirements of the users. Since it does not include any implementation details, it is easier to understand and can be used to communicate with non-technical users. This schema can be defined using existing data models, such as the ER model [33].

A logical schema is produced according to a chosen paradigm, e.g., relational, object-oriented, or object-relational. Therefore, a set of mapping rules that allows to transform a conceptual schema into a logical schema is required. For example, a conceptual schema based on the ER model can be transformed into relational databases applying well-known mapping rules, e.g., [48].

Finally, a definition of a physical schema includes the internal data structures, file organization, indexes, among others. The physical schema is highly technical and considers specific features of commercial DBMSs in order to improve storage and performance.

In the database community, it has been acknowledged for several decades that conceptual models allow a better communication between designers and users in order to understand application requirements. A conceptual schema is more stable than implementation-oriented (logical) schemas, which must be changed whenever the target platform changes [158]. Conceptual models also provide better support for visual user interfaces, for example, ER models have been very successful with users [158].

There have been several proposals for conceptual multidimensional modeling based on the UML notation (e.g., [4, 115]), on the ER model (e.g., [183, 199]), or using specific notations (e.g., [61, 82, 174, 201]). These models include features specific for DW applications, such as dimensions, hierarchies, and measures. However, they have several drawbacks, to which we will refer in Section 1.2.1 and in more detail in Section 2.6.

1.1.5 Methodology for database and data warehouse design

The methodology for the design of operational or transactional databases has well-defined phases of requirement specifications, conceptual, logical, and physical modeling. The requirement gathering process results in a concisely written specification of operational users’ demands. In the next step this information is used for creating a conceptual schema mostly based on the ER or UML models. The subsequent two steps transform the conceptual schema into the logical one, e.g., relational, which is later on implemented using a specific DBMS.

Since DWs are databases aiming at supporting the decision-making process, the phases used in conventional database design have also been adopted for developing
1.2. MOTIVATION AND OBJECTIVES OF THE THESIS

During the requirement gathering process users at different levels of management are interviewed to find their analysis needs. The obtained specification serves as basis for creating a database schema able to respond to users' queries. In many situations due to the lack of a well-accepted conceptual model for DW applications, designers skip the conceptual modeling phase using instead the logical representation based on so-called star and/or snowflake schemas. Afterwards, the physical modeling considers the facilities provided by current DBMSs.

However, many real DW projects show that the development of DW systems is rather different than the development of conventional database systems. Therefore, some modifications to the above-described methodology are necessary. For example, unlike conventional databases, the DW data is extracted from source systems, thus not only users' demands but also data availability are important to consider. Non-traditional approaches can also be used for designing DWs; they mainly take into account the underlying operational databases instead of relying on users' demands. Additionally, the data extracted from source systems in many cases must be transformed before being loaded into the DW. Therefore, it is necessary to consider the so-called extraction-transformation-loading (ETL) process during the DW design.

The design of spatially- or temporally-extended databases does not have a well-established methodology. The usual practice is to design conventional databases ignoring spatial and temporal aspects, and later on, to include elements that allow to represent these kinds of support. A similar situation arises for spatial and temporal DWs with the additional difficulty that being a very recent research area, less experience is available for developing them.

1.2 Motivation and objectives of the thesis

In this section, we present the rationale that motivated us to explore the field of conceptual modeling for DW and OLAP applications. We also refer to the importance of the inclusion of spatial and temporal data in a conceptual multidimensional model. Finally, we describe different aspects that motivate us to propose a methodology for the design of conventional, spatial, and temporal DWs.

1.2.1 Motivation

The domain of conceptual design for DW applications is still at a research stage. The analysis presented by Rizzi [180] shows the little interest of the research community in conceptual multidimensional modeling. Even though conceptual models are closer to the way users perceive an application domain than logical models, the current state of affairs is that logical models are used for designing DWs. Therefore, when building DWs, users have difficulties in expressing their requirements since specialized knowledge related to technical issues is required. Further, logical models limit users to define only those elements that the underlying implementation systems can manage, for example, they may allow users to define only simple hierarchies.

Even though there are some conceptual models for DW design, they either provide a graphical representation based on the ER model or the UML notation with little or
no formal definition, or only provide a formal definition without any graphical support. They also do not include all the necessary components that allow to express analysis needs in unambiguous way; for example, to distinguish different kinds of hierarchies existing in real-world applications. This situation is considered as a shortcoming of existing models for DWs [79].

Furthermore, although location information, for example, address, city, or state, is included in many DWs, it is usually represented in an alphanumeric manner. Replacing this alphanumeric data with spatial data leads to the emerging field of spatial DWs. However, this field raises several research issues.

Firstly, there is no consensus in the literature about the meaning of spatial DWs. The term is used in different situations, for example, when there are high volumes of spatial data, when the integration or aggregation processes for spatial data are required, or when the decision-making process uses spatial data. Even though it is important to consider all the above aspects in spatial DWs, what is still missing is a proposal for spatial DWs with clearly-distinguished spatial elements required for multidimensional modeling. This proposal should consider that although both, spatial DBs and spatial DWs, manage spatial data, their purpose is different. Spatial DBs are used for answering queries that involve spatial location. Examples of such queries are where is the closest store to my house, which highways connect Brussels and Warsaw, how to get to a specific place from my current position given by a GPS. In contrast, spatial DWs use only the data of spatial DBs that is needed to support the decision-making process, for example, what is the best location for a new store, which alternative routes can be built for highways with intensive traffic.

Secondly, applications that include spatial data are usually complex and need to be modeled taking into account user requirements. However, the lack of a conceptual approach for DW and OLAP systems joined with the absence of a commonly-accepted conceptual model for spatial applications, make difficult the modeling task. Usually the systems are firstly implemented and later on, if user expectations are not satisfied, some approach that allows to better understand user requirements is used. Further, although some existing conceptual models for spatial DBs are able to represent relationships among spatial objects, they are not adequate for multidimensional modeling since they do not include the concepts of dimension, hierarchy, measure. To the best of our knowledge, there is no proposal to exploit the possibility of conceptual modeling of spatial data for DW applications.

Finally, a proposal for a spatially-extended conceptual multidimensional model should consider different aspects not present in conventional multidimensional models, such as topological relationships, aggregations of spatial measures, among others. Some of these aspects are briefly mentioned in the existing literature (for example, spatial aggregations), others are neglected (for example, the influence of topological relationships between spatial objects forming hierarchies on aggregation procedures).

Another important characteristic of DWs is that data is stored for long periods of time. According to the DW features of “time-variant” and “non-volatility” (Section 1.1.1), changes to data cannot overwrite the already existing values. However, usual multidimensional models are not symmetric in the way of representing changes for measures and dimensions: while they allow to track changes in measures, they are not able to represent changes in dimension data and the time when these changes have
occurred, for example, when a product has changed its ingredients. Consequently, the mentioned DW features only apply for measures leaving to applications the representation of changes occurring in dimensions.

Currently, to represent changes in dimension data, several implementation solutions were proposed for relational databases for so-called *slowly-changing dimensions* [100]. The first solution, called *type 1 or the overwrite model*, replaces old dimension records with the new data loosing the track of data changes that is needed for analysis. The second solution, called *type 2 or the conserving history model*, inserts a new record containing different values for those attributes, for which changes were applied. However, this approach is complex, since it requires the generation of keys for including different records that represent the same entity only with changes to some attribute values. It introduces unnecessary data repetition for the values that do not change in time and demands significant programming effort for querying time-varying dimension data. The last solution, called *type 3 or limited history model*, allocates several additional fields for every attribute, for which changes in values are important to keep. These fields store the old and new values of the attribute. Even though the implementation is trivial, the history of changes is limited to the number of additional fields.

As can be seen, proposed solutions are not satisfactory since they either do not preserve the entire history of data or are complex for the implementation. Further, they do not consider research related to managing time-varying information in temporal databases. Therefore, it is expected that temporal DWs provide solutions that help to solve problems related to the lack of symmetry in multidimensional models.

Currently, the research related to temporal DWs raises many issues mostly dealing with implementation aspects, such as among others, special aggregation procedures, special storage and indexing methods. However, very little attention from the research community has been drawn to conceptual modeling for temporal DWs and to the analysis of which temporal support should be included in these systems. This analysis should consider that temporal DBs and conventional DWs have some similarities, such as both manage historical data, however they are semantically different. For example, data in DWs is integrated from existing source systems whereas data in temporal DBs is introduced by users since it represents operational or transactional databases; DWs support the decision-making process while temporal DBs reflect data changes in the reality and in the database content; DW data is neither modified nor deleted^2, in contrast, users of temporal DBs change data directly. Therefore, we consider that it is necessary to propose temporal support adequate for DW applications and a conceptual model with time-varying elements that takes into account specific semantics of DWs.

The development of conventional, spatial as well as temporal DWs is a very complex task that includes many phases for its realization. In order to assist developers and to facilitate the execution of different phases, the specification of a methodology is required.

Existing methodologies offer a variety of solutions especially for the requirements specification phase. However, such diversity of approaches can be confusing for designers. For example, one approach first considers users’ requirements and then checks them against data availability in source systems while another approach first devel-

---

^2We ignore modifications due to errors during data loading and deletion for purging DW data.
ops the DW schema based on the underlying source systems and then adapts it to users’ requirements. Therefore, it is not clear in which situations it is better to use each of them. As a consequence many DW projects do not give much importance to the requirements specifications and focus on more technical aspects [153]. This may lead to the situation where the DW may not meet users’ expectations to support the decision-making process.

In addition, some of the existing proposals include additional phases according to particular characteristics of the applied conceptual multidimensional model. In this way, these methodologies force designers to use a conceptual model that is not known by them and may also be not required for the particular DW project.

Moreover, even though many organizations lack tactic and strategic information and feel attracted to the DW technology, they might not afford external teams or consultants to build a DW. Nevertheless, they have professionals highly skilled in the development of traditional transactional or operational databases, but inexperienced in the development of DWs. Therefore, providing a methodological framework for DW development may facilitate the transition process from the development of traditional databases into the development of DWs.

Finally, since spatial and temporal DWs are new research fields, there is not yet a methodological approach for developing them. However, it is well-known that the inclusion of spatial data empowers decision-making processes providing visualization facilities and expanding analysis. Similar to spatial data, temporal support allows to expand analysis by relying on time-varying data in all elements of a multidimensional model. Therefore, the proposed methodology should specify how spatial and temporal support may be included considering all the phases required for the development of spatial and temporal DWs.

1.2.2 Objectives of the thesis

**General objective** To define a multidimensional conceptual model and an associated design methodology able to express data requirements for data warehouse applications with conventional, spatial, and time-varying data.

**Specific objectives**

- Establish and define the necessary elements for conceptual modeling of conventional data warehouses. This is the subject of Chapter 2.

- Include spatial extensions for the conceptual model for data warehouses considering specific characteristics of a multidimensional model and spatial data. We refer to this objective in Chapter 3.

- Extend a conceptual model for data warehouses allowing to represent changes to data based on the concepts used in temporal databases. Chapter 4 is dedicated to this subject.

- Propose mappings of conventional, spatially-extended, and temporally-extended multidimensional models to logical models. This subject is treated in Chapters 2, 3, and 4.
1.3 Scope and contributions of the thesis

This thesis summarizes an original research conducted by the author, which has been partly published in international journals and conference proceedings [116, 117, 118, 120, 119, 121, 122, 123, 124].

In this thesis, we propose the MultiDimER model – a conceptual model able to represent data requirements for conventional, spatial, and temporal DW applications. The model includes graphical notations and formal definition.

1.3.1 Conventional data warehouses

The main contribution of our proposal for conventional DWs refers to several aspects:

- The MultiDimER model is based on the well-known ER constructs of entity types, relationship types, and attributes with their usual semantics. Therefore, it allows designers to use the same modeling constructs as those used for operational database design and it provides a conceptual representation of data independent of technical details.

- The model considers a multidimensional view of data making possible to distinguish different elements such as dimensions, hierarchies, and measures. In this way, it allows to better represent data required for DW and OLAP applications.

- Our proposal extends the semantics of the MultiDimER model by the inclusion of different kinds of hierarchies existing in real-world applications. We classify these hierarchies according to the relationships existing between them.

Since current DW and OLAP systems allow to manage only the limited number of hierarchies, our proposal has important implications for users as well as for designers and implementers. It allows a clear distinction of each type of hierarchy taking into account their differences at the schema as well as at the instance levels. Therefore, it gives users a better understanding of the data to be analyzed. It also helps designers to build conceptual models for DW applications representing with clarity different kinds of hierarchies. Additionally, the proposed classification provides OLAP tool implementers the requirements needed by business users for extending the functionality of current OLAP tools.

- The proposal includes the mapping of different kinds of hierarchies to relational databases based on well-known mapping rules (e.g., [48]). In some cases, we modify these rules in order to express in a better way the semantics of a specific hierarchy. Therefore, we ensure that different kinds of hierarchies can be implemented in current DBMSs, in spite of considering some of them as advanced features of a multidimensional model [197].
CHAPTER 1. INTRODUCTION

More details related to the above-specified contributions can be found in Chapter 2 as well as in the following articles:


Additional contributions refer to the methodological framework proposed for conventional DW design:

- In spite of the existing similarity between database and data warehouse design phases, we modify their description and include additional phases according to specific features of DWs. The proposed methodology for DW design gives the DW developer team insights of the required design phases and their content. It allows to obtain the correct comprehension of user requirements, a fine-tuned design phase, and the construction of multidimensional schema that enables decision makers to perform all necessary analysis.

- Based on a variety of existing solutions for conventional DW design, we propose and exemplify three different approaches that can be used for requirements specifications leading to a conceptual schema. For each of them, we give general and schematic descriptions. We include guidelines stating in which situations each approach may give better results. These recommendations refer to characterization of users, of a developer team, and of source systems as well as to general advantages and disadvantages.

- Considering that DWs use data from source systems, we include an additional phase that concerns data extraction, transformation, and loading processes. This phase forms part of the conceptual, logical, and physical design. In this way, we allow implementers to first define and refine such as processes without being burdened with implementation details of the underlying database management systems and later on, implement them.

- Taking into account that proposed methodology does not depend on conceptual model, on target implementation platform, or on database logical and physical organization, it can be useful for experienced DW developers as well as for operational database developers who want to acquire basic knowledge for developing a DW. Experienced DW developers can benefit from many novelties included in the
1.3. SCOPE AND CONTRIBUTIONS OF THE THESIS

requirements specifications that may be not known by practitioners. The latter may gain advantages in the learning process due to the systematic representation of different phases and methods supported with simple examples that show their applicability.

The more detailed description of our proposal related to methodology for conventional DW design can be found in Chapter 5.

1.3.2 Spatial data warehouses

We extend the MultiDimER model allowing to include spatial data in dimensions, hierarchies, and measures. The novelty of our approach consists in the following:

- The multidimensional model is seldom used for representing spatial data even though this model provides a concise and organized representation for spatial DWs [12] and facilitates the delivery of data for spatial OLAP systems. Since the proposed model is platform independent, it assists in a better way the design process in contrast to current data models used for spatial applications. The latter often refer to particular aspects of logical-level design and are too complex to be understood by users.

- Our proposal allows to include spatial support for different elements of the multidimensional model, i.e., dimensions, hierarchies, and measures. We formally defines each element and refer to particular characteristics of each of them. In this way, users, designers, and implementers can clearly distinguish every element and focus on important aspects related to their management.

- The model includes a new feature, the so-called spatial fact relationship, that allows to represent the topological relationships existing between spatial dimensions that are of users' interest.

- The proposal extends the previously-defined classification for non-spatial hierarchies by inclusion of different types of spatial hierarchies. Since spatial objects forming a hierarchy are topologically related, we classify the complexity of required aggregation procedures considering these topological relationships. This classification gives insights for spatial OLAP implementers indicating when the aggregation of measures can be done safely and determining when special aggregation procedures must be developed.

- This thesis also refers to mapping of a spatially-extended conceptual multidimensional model to an object-relational representation. In this way we show that schemas created using our model can be implemented in general-purpose DBMSs that include spatial extensions. We also specify spatial integrity constraints required for preserving semantics of a conceptual model during the transformation to a logical model. We demonstrate how the spatial operators and aggregation functions existing in current DBMSs can be used for typical OLAP operations including aggregation procedures.
• Our proposal extends methodology for conventional DW design by inclusion of spatial data in different elements composing the multidimensional model. This methodology differs from methodology proposed for spatial database design since it considers whether users have knowledge in spatial data analysis and manipulation and whether source systems include spatial data.

Chapters 3 and 5 as well as the following articles are dedicated to the above mentioned subjects:


1.3.3 Temporal data warehouses

Additionally, we propose the inclusion of temporal support for the MultiDimER model. Similarly to the spatial extension, we consider different elements composing the multidimensional model. The proposed temporal extension contributes in several ways to the field of temporal DWs:

• We include in the MultiDimER model different temporal types that may be required for DW applications and that are currently ignored in research proposals. We based our proposal on research achievements in temporal databases and on considering semantic differences between temporal DWs and temporal databases. We also take into account the specific characteristics of multidimensional models.

• We extend the model including temporal support for dimensions, hierarchies, and measures. Therefore, we incorporate temporal semantics as an integral part of a conceptual multidimensional model. We believe that in this way we can help users and designers to choose and to express in an unambiguous way which elements of DWs they want to be time invariant and for which data they want to express changes occurred in time. We also consider that the proposed temporal extension gives a symmetry to the multidimensional models allowing to express in the same way changes to dimension data as well as to measures that currently cannot be done.
1.4. THESIS ORGANIZATION

- We establish conditions for including temporal support in hierarchies. In this way, we help implementers to develop aggregation procedures that ensure correct management of hierarchies in DW and OLAP applications.

- We define temporal measures changing the current role of the time dimension, i.e., this dimension is not required anymore to indicate the validity of the measures. We also show the usefulness of having different temporal types for measures extending the analysis spectrum. Currently, this aspect is ignored in research related to temporal DWs allowing only the presence of valid time. Additionally, we propose the inclusion of different temporal support depending on whether or not measures are aggregated before loading into temporal DWs.

- We refer to problems related to multiple time granularities between source systems and temporal DWs. The proposed solutions demonstrate the relevance of the research in the area of temporal databases for temporal DWs.

- Similarly to the previous cases, we propose a mapping to a logical model. Since there are no DBMSs that are temporally enhanced even though some proposals exist (e.g., [39, 189]), we use object-relational databases. This mapping allows to assist implementers who use the MultiDimER model for conceptual design of temporal DWs.

- Finally, we propose the methodological framework for temporal DW design extending the methodology described for conventional DW design. We refer to different cases considering users' requirements for having temporal support and availability of historical data in source systems.

More details about our contribution can be found in Chapters 4 and 5 as well as in the following articles:


1.4 Thesis organization

The thesis is organized in several chapters, most of which are partially based on the previously-mentioned articles.

Chapter 2 defines the MultiDimER model, including the different kinds of hierarchies as well as their classification. For each of them graphical notations and mapping to relational databases are given. This chapter is based on Articles 1 and 2.
Chapter 3 refers to the spatial extension of the MultiDimER model considering both the conceptual as well as the logical levels. By means of examples, we discuss the inclusion of spatial characteristics for the different elements forming a multidimensional model, such as dimensions, hierarchies, and measures. Further, in this chapter we study and classify topological relationships existing between spatial elements forming a hierarchy indicating when aggregations of measures can be done safely and when they require special handling. We also present how the semantics can be preserved when transforming a conceptual model to a logical model. This chapter is partially based on Articles 3, 4, and 5.

Chapter 4 presents the MultiDimER model with its temporal extension. We propose the inclusion of different temporal support required for analysis purposes. Then, taking into account that dimensions and measures play different roles in multidimensional models, we discuss the aspects related to inclusion of temporality for each of these elements of a multidimensional model. Along the proposal of this chapter, we present conceptual model and its logical representation. This chapter is based on Articles 7, 8, and 9.

Chapter 5 includes the specification of the methodology for conventional, spatial, and temporal DW design. Our proposal is in line with the traditional database design phases. For the requirements specification phase we provide three different methods giving their descriptions and recommendations when to use them. For each method we give examples to facilitate their comprehension and to show their applicability in real-world situations. The requirements specification process finishes with the conceptual multidimensional model. Then, we refer to logical and physical level design considering DW structures and applications as well as extraction-transformation-loading (ETL) processes. This chapter is partially based on Article 6.

Chapter 6 summarizes the results contained in the thesis and indicates some directions for future research.

Appendix A presents the formalization of the MultiDimER model. It defines syntax, semantics, and semantic constraints. This formalization was included in Article 2.

Appendix B and C present the formalizations of the spatially- and temporally-extended MultiDimER model, respectively. It defines the syntax as well as the semantics of the model, including their semantic constraints.

Appendix D includes the notations used in this thesis for representing the constructs of the ER, relational, and object-relational models as well as conventional, spatial, and temporal data warehouses.
Chapter 2

A conceptual model for conventional data warehouses

Data warehouses include considerable amount of historical data in order to help users on the management levels of organizations to make more effective and better decisions. In order to express in a better way users’ data requirement, the usual approach in databases is to use a conceptual model. The advantages of using conceptual models for database design are already well-known. Since data warehouses are databases with the purpose of supporting the decision-making process, they also should be conceptually represented.

In order to help users, designers as well as implementers of DW and OLAP applications, in this chapter we formally define the MultiDimER model, a conceptual multidimensional model able to represent data requirements for DW and OLAP applications, i.e., the model is based on the concepts of dimensions, hierarchies, and facts with associated measures. Since hierarchies are needed in order to exploit DW and OLAP systems to their full capabilities, we consider different kinds of hierarchies existing in real-world situations and classify them giving also their graphical representation. We also propose a mapping of the model to the relational model. We introduce new insights of the relational representation of the different kinds of hierarchies showing the feasibility of their implementation within current database management systems. Since some kinds of hierarchies can be implemented in several ways, we also compare in which situations different mappings work better.

This chapter is organized as follows. Section 2.1 introduces currently existing approaches for DW modeling and refers to the aspects of data manipulation. Section 2.2 presents the MultiDimER conceptual model while Section 2.3 describes general mapping rules from our model to the relational model. Section 2.4 refers to different kinds of hierarchies. For each of them, it includes conceptual representation and presents mapping to the relational databases, refining the mapping rules given in Section 2.3, if necessary. We also discuss existing relational implementations of some of these hierarchies and show in particular how they can be implemented in Microsoft Analysis Services provided with SQL Server 2000 [135]. Section 2.5 shows an application example using the MultiDimER model that includes several fact relationships and hierarchies. Finally, Section 2.6 refers to works related to representing hierarchies in multidimensional models, and Section 2.7 gives summary of this chapter.
2.1 Data warehouse structures and data manipulation

Data warehouse users require a small number of complex queries involving large amounts of data, which usually are summarized (for example, sales during 2005 in the region of Brussels). These queries differ from usual transaction processing applications, which deal with a large number of relatively simple transactions accessing few data items. Therefore, DWs may use different data structures that facilitate formulations and executions of complex queries.

2.1.1 Star, snowflake, and starflake schemas

The structure of data warehouses (DWs) can be represented at a logical level using the so-called star, snowflake, or starflake schemas. These schemas include relational tables termed fact and dimension tables (Figure 2.1). A fact table (for example, Sales facts in Figure 2.1) represents the subject orientation and the focus of analysis, such as analysis of sales. It usually contains numeric data called measures (for example, Quantity or Sales in Figure 2.1) representing analysis needs in quantified form. Dimensions (for example, Product, Time, and Store in Figure 2.1) are used for exploring the measures from different analysis perspectives, such as sales of a specific product in a specific time or sales of all products in a specific store. They often include hierarchies (for example, Product–Category–Department in Figure 2.1). Hierarchies allow to see measures at different levels of details. For example, "moving" from Product to Category (Figure 2.1) gives aggregated measures representing the sum of sales and quantities of products belonging to corresponding categories. The level of details on which a measure is represented is called data granularity. For example, the aggregated measures for the Category level have a coarser data granularity than measures included in the fact relationship for the Product level. Further, a dimension may also have descriptive attributes (for example, Store number or Manager name in the Store dimension). These attributes can give additional information required by users.

The difference between star and snowflake schemas consists in the fact that the latter includes only normalized tables for representing hierarchies, such as Product–Category–Department in Figure 2.1, while the former would have only de-normalized tables. For example, the Store dimension in Figure 2.1 includes a hierarchy consisting of Store–City–State represented as a de-normalized table. The starflake schema combines the representation of hierarchies as normalized and de-normalized tables and this kind of schema is shown in Figure 2.1.

2.1.2 Aspects of data manipulation in DWs

Data contained in a DW can be dynamically manipulated using on-line analytical processing (OLAP) systems. Two types of storage are used for OLAP systems: relational and multidimensional. The former is based on star or snowflake schemas using

---

1 To represent relational tables we use the notations described in Section D.2.
2 They can also be used as stand-alone systems.
2.1. DATA WAREHOUSE STRUCTURES AND DATA MANIPULATION

Figure 2.1: An example of a starflake schema.

relational databases. The latter is usually represented as a vendor-specific array-like structure. In both cases, in order to exploit OLAP systems to their full capabilities, dimensions, measures, and hierarchies must be clearly defined. Hierarchies allow to establish traversing paths to obtain measures on different levels of details using the roll-up and drill-down operations. The roll-up operations automatically transform detailed measures into summarized data according to a specific aggregation function (e.g., sum, average). For example, while traversing from Store to City in a hierarchy included in the Store dimension (Figure 2.1), the measures representing Quantity and Sales for every store will be summarized and grouped for corresponding city. The drill-down operations do the opposite to the roll-up operations, i.e., they allow to transform summarized (aggregated) measures into a more detailed form. Further, the slice-and-dice operations allow to select a portion of the data based on specified values in one or several dimensions, for example, sales of milk during May.

Nevertheless, the types of hierarchies managed by current OLAP tools are very restrictive when modeling real-world situations. One reason for that is that such tools are based on logical and physical models where implementation considerations are mixed with conceptual representations of data. Further, underlying implementation restricts the OLAP systems to include only those hierarchies for which summarizability conditions hold [111].

Summarizability refers to the correct aggregation of measures in a higher hierarchy level (for example, Department in Figure 2.1) taking into account existing aggregations in a lower hierarchy level (for example, Category in Figure 2.1). Summarizability conditions indicate:
1. Disjointness of instances, i.e., grouping of instances according to the next hierarchy level must give disjoint subsets. For example, the same category cannot belong to two departments in the example of Figure 2.1.

2. Completeness, i.e., all instances are included and each instance is related to some instance in the next hierarchy level. For example, in Figure 2.1 all categories are included in the model and each category is assigned to some department.

3. Correct use of statistical functions according to the measure type.

Measures can be of type flow, stock, and value-per-unit [111]. The flow type indicates that the measure is recorded at the end of some period and it records the cumulative effect over a period, such as monthly sales. The stock type refers to the measure recorded at a particular time point and it represents the state at this time point, such as inventory of products. The value-per-unit is similar to the stock type, however the units are different, for example, item price versus amount of total inventory for value-per-unit and stock types, respectively [111]. These measure types determine which statistical functions can be used; for example, monthly sales can be summed over a year, however, the inventory of products cannot be summed over different periods of time.

However, many real-world hierarchies do not satisfy these summarizability conditions. For example, while the geographical division of a country may consist in the hierarchy of City–County–State, some states may not have counties. Therefore, this hierarchy does not satisfy the condition of completeness, since not all cities map to the next hierarchy level, i.e., to County.

### 2.2 MultiDimER: a conceptual multidimensional model

Conceptual models, being platform independent, offer several advantages. For example, they facilitate the communication between users and designers since they do not require the knowledge of specific features of the underlying DBMS. Further, schemas developed using conceptual models can be mapped to logical schemas based on different models, such as relational, object-relational, or object-oriented. Moreover, conceptual models facilitate future extensions since they can be modified in a simple way providing a better support for the subsequent changes in the logical schemas. Therefore, instead of modeling data required for DW and OLAP applications at a logical level (i.e., using relational tables with foreign keys as in Figure 2.1), we propose the MultiDimER model - a conceptual multidimensional model. The last term indicates that our model allows to represent at the conceptual level all elements required in DW and OLAP applications, i.e., dimensions, hierarchies, and facts with associated measures.
2.2. THE MULTIDIMER MODEL

2.2.1 Model definition

The MultiDimER model uses the ER-like graphical notations shown in Figure 2.2\(^3\). In Section 2.2.3 we present a metamodel of our conceptual model and Appendix A includes its formal definition.

![Level name](image)

![Key attribute](image)

![Other attributes](image)

![a)](image)

![b)](image)

![c)](image)

![d)](image)

![e)](image)

Figure 2.2: Notations for multidimensional model: a) level, b) hierarchy, c) analysis criterion, d) cardinalities, and e) fact relationship with associated measures.

The MultiDimER model is a finite set of dimensions and fact relationships. A dimension is an abstract concept for grouping data that shares a common semantic meaning within the domain being modeled. It represents either a level, or one or more hierarchies. Levels correspond to entity types (Figure 2.2 a)). Every instance of a level is called member.

Hierarchies are required for establishing meaningful paths for roll-up and drill-down operations. A hierarchy contains several related levels. Since the binary relationship linking the levels of a hierarchy is only used for traversing from one level to the next one during the roll-up and drill-down operations, we do not include a special symbol for indicating it, simplifying in this way the notation as shown in Figure 2.2 b). Hierarchies express different structures according to an analysis criterion (Figure 2.2 c)), for example, geographical location or organizational structure.

Given two consecutive levels of a hierarchy, the lower level is called child (the level with name\(_1\) in Figure 2.2 b)) and the higher level is called parent (the level with name\(_2\) in Figure 2.2 b)). A level of a hierarchy that does not have a child level is called leaf; the last level, i.e., the one does not have a parent level is called root. The root represents the most general view of data. Even though in some works the root of a hierarchy is represented using a level called ALL, we consider that its inclusion in a conceptual model can be ambiguous or meaningless for decision-making users.

The relationships between child and parent levels are characterized by cardinalities. Cardinalities (Figure 2.2 d)) indicate the minimum and the maximum numbers of

\(^3\)More detailed description of our notation is given in Section D.3.
members in one level that can be related to a member in another level.

Levels contain one or several key attributes (underscored in Figure 2.2) and may also have other descriptive attributes. Key attributes of a parent level define how child members are grouped. Key attributes in a leaf level or in a level forming a dimension without hierarchy indicate the data granularity of measures in the fact relationship. Notice that the notation in Figure 2.2 allows to group the attributes into their corresponding levels enriching the expression power of the model.

A fact relationship (Figure 2.2 e)) represents an n-ary relationship between leaf levels. It expresses the focus of analysis and may contain attributes commonly called measures. The latter usually represent numerical data meaningful for leaf members that are aggregated while traversing a hierarchy. Since a cardinality ratio for every leaf member participating in a fact relationship is always (0,N), we omit such cardinalities to simplify the model.

Figure 2.3 illustrates the conceptual schema of the Sales DW\(^4\) commonly represented using relational tables as was shown in Figure 2.1. This schema preserves the characteristics of star, snowflake, or starflake schemas providing at the same time a more abstract conceptual representation.

2.2.2 Model particularities

The MultiDimER model is mainly based on the existing ER constructs, for example, entity types, attributes, relationship types. Additionally, our model offers explicit

\(^4\)The Time and Store dimensions may also have hierarchies, but they are not represented for simplicity.
support for representing different kinds of hierarchies. Currently, we do not consider other ER constructs, such as generalization, multivalued attributes, and composite attributes. The inclusion of these features is not straightforward and requires analysis of their usefulness in multidimensional modeling.

Some important features of MultiDimER model should be emphasized. For example, the levels can be shared between different hierarchies and the same leaf level can participate in different fact relationships avoiding unnecessary repetition of levels and hierarchies. Further, the MultiDimER model allows to define several child-parent links between the same levels as we will see in Section 2.5.

Notice that unlike [2] we do not consider hierarchies as a part-whole relationship (e.g., [23, 76]) since this may be semantically incorrect. For example, in a hierarchy Client–City–State, while cities are part of a state, it is not meaningful to consider that clients are part of a city.

Further, Abelló et al. [2] consider child-parent relationships as collections, i.e., as part-whole relationships when a whole is composed by parts having a uniform compositional structure. This condition is too restrictive for some kinds of hierarchies, i.e., for generalized and non-covering hierarchies (Section 2.4.1.3). Moreover, if in addition to the hierarchy Client–City–State another hierarchy, for example, Store–City–State is created (as in Figure 2.26), stores as well as clients will be taken as parts of a city. Therefore, a whole represented by the city would be composed by semantically different and heterogeneously structured parts, i.e., clients and stores.

2.2.3 Metamodel of the MultiDimER model

In this section we present the metamodel of the MultiDimER model as shown in Figure 2.4. A dimension is composed of either a level, or one or more hierarchies. Dimension's name is derived from the leaf-level's name. A hierarchy can belong to only one dimension. A hierarchy contains several related levels and these levels may be shared between different hierarchies. A Criterion is used for identifying a hierarchy.

Each level has a unique name. Levels include attributes, some of which are key attributes used for aggregation purposes while others are descriptive. Levels forming a hierarchy are associated through the child-parent relationship (the Connects relationship in Figure 2.4). This relationship is characterized by the minimum and maximum cardinalities expressed in the child and parent roles.

A fact relationship represents an n-ary relationship between levels with $n > 1$. The fact relationship may contain attributes representing the so-called measures.

2.3 Mapping from conceptual to relational models

In this section, we present mapping rules that allow to implement the conceptual schemas designed using the MultiDimER model in relational database management systems.
2.3.1 Rationale for choosing relational model

Conceptual models can be implemented using different logical models. For operational database systems the most common practice is to represent a conceptual schema using the ER model and then transform it to, for example, a relational schema. The latter is highly normalized in order to avoid update anomalies [48]. Since DWs are databases with the purpose of supporting the decision-making process, they also can be conceptually represented using, for example, our model and then implemented using different logical models, such as relational, object-relational. We have chosen the relational approach for implementing a multidimensional model for the following reasons:

- DWs are usually implemented in relational databases as star and snowflake schemas as was already explained in Section 2.1. They mostly use OLAP systems to manipulate data through roll-up and drill-down operations.

- Relational databases use well-known strategies of data storage, are well standardized, tool independent, and readily available. In contrast, there is no consensus in the research community for defining a logical level for OLAP systems [6] and additionally, multidimensional OLAP (MOLAP) systems differ greatly in the structures used for data storage.
2.3. MAPPING FROM CONCEPTUAL TO RELATIONAL MODELS

- Much research has been done in relational databases for improving optimization techniques, indexing, join operations, and view materialization considering particularities of DW applications.

- The relational model allows a logical-level representation of the different kinds of hierarchies. This would facilitate the subsequent development of MOLAP systems according to the specific data storage structures.

- Even though MOLAP systems are frequently used for storage purposes since they offer better performance during the roll-up operations, many commercial and prototype systems use ROLAP (relational OLAP) and HOLAP (Hybrid OLAP: a combination of ROLAP and MOLAP systems) systems when analysis of large volumes of more detailed data is required.

- Commercial relational database systems [135, 147] include extensions to represent and manage multidimensional views of data.

- Relational databases are an accepted resource layer in the Common Warehouse Model (CWM) [143] from the Object Management Group (OMG). As implementation model we choose the surrogate-based relational model [66]. In this model each row in a table has an associated surrogate, i.e., a system-generated artificial key. Surrogate values cannot be seen or modified by users. The reason for using surrogates instead of relying on user-defined keys is that DWs usually use system-generated keys for ensuring both, better performance during join operations and independency from transactional systems. Further, surrogates do not vary over time so that two entities having identical surrogates represent the same entity, thus allowing to include historical data in an unambiguous way. In addition, in some situations it is necessary to store the information about the entity either before it has been assigned a user-controlled key value (for example, a student is introduced in a database before assigning him an ID number) or after it has ceased to have one (for example, an employee left an organization). Further, the SQL:2003 standard [130, 131] as well as some current DBMSs (for example, Oracle 10g [149]) include object-relational extensions that allow to have tables with rows containing automatically-created surrogates. Nevertheless, the relational model without surrogates can also be used in our mapping for DBMSs that do not allow surrogates. In this case, implementers are responsible for creating keys independent from those used in source systems.

2.3.2 Mapping rules

Since the MultiDimER model is based on the ER model, its mapping to the relational model is based on well-known rules described, for example, in [48, 10]:

1. A level type corresponds to a regular entity type in the ER model. It maps to a relation containing all its attributes and including an additional attribute for a surrogate key.

5See more description about CWM in Section 2.6.
2. A relationship type between child and parent levels of a hierarchy corresponds to a binary relationship type in the ER model. Two different mappings exist depending on the cardinality of the child role since in order to have meaningful hierarchies, we suppose that the maximum cardinality of the parent role is always N:

(a) If the cardinality of the child role is (0,1) or (1,1), i.e., the cardinality between child and parent levels is many-to-one, the relation corresponding to the child level is extended with the surrogate key of the corresponding parent level (i.e., there is a foreign key in the child table coming from its parent table).

(b) If the cardinality of the child role is (0,N) or (1,N), i.e., the cardinality between child and parent levels is many-to-many, a new relation is created that contains as attributes the surrogate keys of the child and parent levels.

3. A fact relationship type corresponds to an n-ary relationship type in the ER model. For each fact relationship type involving leaf levels \( L_1, L_2, \ldots, L_n \), a new relation is created that includes as attributes the surrogate keys of the participating levels. Additionally, the relation includes the attributes of the fact relationship type.

An example of this mapping is presented in Figure 2.1 for the MultiDimER schema of Figure 2.3. Notice that underscored attributes in the MultiDimER model indicate attributes used for aggregation purposes while in the relational model underscored attributes represent unique identifiers for members, i.e., as primary keys.

Even though the proposed mapping works well in many cases, we will revise it taking into account (1) the different kinds of hierarchies defined in the next section, (2) the fact that hierarchies are only used for traversing from a detailed to a more general level or vice-versa, and (3) already existing support in current DWs and ROLAP (relational OLAP) systems for representing some kinds of hierarchies. For the latter, we consider useful to indicate when it is more practical to apply different representations.

2.4 Data warehouse hierarchies

In real-world situations users deal with different kinds of hierarchies that are difficult to represent using current DW and OLAP logical models. Therefore, users often are not able to capture the multidimensional semantics of applications and must limit their analysis considering only a predefined set of hierarchies. Further, current approaches in conceptual representations of hierarchies focus mainly on aggregations paths represented at the schema level, i.e., establishing sequences of levels that should be traversed during the roll-up and drill-down operations. However, as we will see in this section, the distinction between different kinds of hierarchies should be made also at the instance level, i.e., considering cardinalities existing between the child and parent levels.

We present next the categorization of hierarchies. We distinguish two types of hierarchies, simple and multiple, where the latter are composed of one or several simple hierarchies accounting for the same analysis criterion. Moreover, simple hierarchies
include further types: symmetric, asymmetric, and generalized hierarchies. Also, non-covering hierarchies are considered as a special case of generalized hierarchies. For each of these simple hierarchies, another specialization can be applied depending on whether the cardinalities between the child and parent levels are many-to-one or many-to-many. The former is called strict hierarchies while the latter non-strict hierarchies. Finally, parallel hierarchies are obtained when several hierarchies are associated to the same dimension, each one accounting for different analysis criteria. Parallel hierarchies are further specialized in independent or dependent hierarchies depending on whether they do not share or share levels, respectively. A summarized representation of the classification of different kinds of hierarchies and relationships existing between them is included in Figure 2.25 of Section 2.4.5.

In the following we refer in more details to different kinds of hierarchies presenting them at conceptual and logical levels.

2.4.1 Simple hierarchies

*Simple hierarchies* are those hierarchies where the relationship between their members can be represented as a tree. Further, these hierarchies use only one criterion for analysis.

2.4.1.1 Symmetric hierarchies

**Conceptual representation.** A *symmetric hierarchy* has at the schema level only one path (Figure 2.5 a)). At the instance level the members form a tree where all the branches have the same length (Figure 2.5 b)). As implied by the cardinalities, all parent members must have at least one child member and a child member cannot belong to more than one parent member. For example, in Figure 2.5 each category has assigned at least one product and the product cannot belong to more than one category.

**Logical representation.** Mapping the symmetric hierarchy of Figure 2.5 a) to the relational model gives the relations presented in Figure 2.1 for the Product dimension\(^6\). This mapping is commonly known as the snowflake schema [101] as we already described in Section 2.1. Another approach called star schema [101] de-normalizes the snowflake representation mapping every dimension to one relation; for example, the Product, Category, and Department levels will be represented in the same relation. Each attribute of every level forming a hierarchy corresponds to an attribute of the relation. Further, an attribute for the surrogate key is included.

The star schema representation has several advantages: the schema is easy to understand for query formulation, it can be easily accessed, and few joins are needed for expressing queries due to the high level of de-normalization. Additionally, much research has been done to improve system performance when processing star queries (e.g., [61, 97]). Finally, this data structure allows to easily aggregate measures using, for example, the SQL group by, rollup, or cube operators. Some disadvantages of this

\(^6\)We use the leaf level name as a dimension name.
CHAPTER 2. CONVENTIONAL DATA WAREHOUSES

representation are as follows: it does not allow to model adequately hierarchies since additional information is required for representing a hierarchy [88]. For example, for the Store dimension in Figure 2.1, it is not clear which attributes can be used for forming hierarchies. As can also be seen in Figure 2.1, it is difficult to clearly associate attributes within their corresponding levels. Moreover, in hierarchies having many levels the number of attributes is at least as large as the hierarchy depth making the hierarchy structure difficult to understand. Due to de-normalization and consequently data redundancy, well-known problems also take place.

On the other hand, the snowflake schema better represents a hierarchical structure and levels can be reused between different hierarchies. Additionally, this representation can easily manage heterogeneity across levels [88], i.e., allowing different levels of a hierarchy to include their specific attributes. For example, the Product, Category, and Department levels in Figure 2.1 have their specific attributes. As for the de-normalized approach, the aggregations of measures are straightforward. Further, in some applications the snowflake schema can improve system performance in spite of requiring join operations between relations representing levels [128]. While the snowflake schema removes some of the shortcomings of the star schema, it still has some limitations. For example, the snowflake schema cannot be used without modifications if there is heterogeneity within a level [88], i.e., different characteristics for a group of members of a level. This situation may occur when different products have their own specific attributes.

At the conceptual level the MultiDimER model allows to include a similar representations to star or snowflake schemas, i.e., a one-level dimension and a dimension having a hierarchy. However, in our model representing a dimension having only one level (such as Time and Store dimensions in Figure 2.3) indicates that the user is not interested in exploring hierarchies. On the other hand, hierarchies should be represented as in Figure 2.5 a) when they are used for exploring measures at different granularities,
2.4. DATA WAREHOUSE HIERARCHIES

i.e., using roll-up and drill-down operations. Afterwards, the corresponding relations can be de-normalized into a star schema taking into account performance issues.

Both the snowflake and the star schema representations are already implemented in commercial products [135, 147]. For example, Microsoft Analysis Services allows to define attributes used for aggregations, called *levels*, and additional descriptive attributes, called *member properties*. Both kinds of attributes must be stored in the same table.

2.4.1.2 Asymmetric hierarchies

**Conceptual representation.** An *asymmetric hierarchy*\(^7\) has only one path at the schema level. However, as implied by the cardinalities, at the instance level, some parent members may not have associated child members. Figure 2.6 a) shows a hierarchy where a bank is composed of several branches: some of them have agencies with ATM, some only agencies, and small branches do not have any organizational division. As a consequence, at the instance level the members represent a non-balanced tree (Figure 2.6 b)), i.e., the branches of the tree have different lengths. As for symmetric hierarchies, the cardinalities imply that every child member may belong to at most one parent member. For example, in Figure 2.6 every agency can only belong to one branch.

**Logical representations.** Since asymmetric hierarchies do not satisfy the summarizability condition [111], the mapping of levels and child-parent relationships described in Section 2.3 may lead to the problem of excluding from the analysis the members of the higher levels that do not have child members. For instance, since in Figure 2.6 a) all measures are associated with the ATM level, these measures will be aggregated into the higher levels only for those agencies that have ATMs and afterwards, only for those branches that have agencies. To avoid this problem two alternative solutions are proposed: (1) transforming an asymmetric hierarchy into a symmetric one, and (2) creating parent-child relations.

Transforming an asymmetric hierarchy into a symmetric one is realized using placeholders (PH in Figure 2.7) [161] or null values [29, 135] in “missing” levels. Afterwards, the logical mapping for symmetric hierarchies may be applied. Although this representation is useful in some applications, it has the following shortcomings:

1. A fact relationship table must include common measures belonging to different hierarchy levels. For example, ATMs, agencies, branches, etc. should have the same measure.

2. Special placeholders must be created and managed adequately for aggregation purposes. For example, repeating the same measure value for branch 2 and two placeholders representing missing levels in Figure 2.7.

3. Unnecessary introduction of meaningless values requires more storage space.

---

\(^7\)Several terms are used for these hierarchies: heterogeneous [80], non-balanced [135], non-onto [161].
4. A special interface should be implemented for hiding placeholders from users.

5. Common measures used for all levels have different data granularity. For example, measures for the ATM level as well as for the Agency level for those members that do not have ATM are included in the same fact relationship table.

Another solution consists in creating parent-child relations\(^8\) [135, 147] as shown in Figure 2.8 a) (the instances in Figure 2.8 c) are those from Figure 2.6 b)). At a conceptual level this can be represented in the MultiDimER model using recursively related levels as shown in Figure 2.8 b).

While a parent-child structure gives a more compact representation than a flat table or a snowflake structure avoiding duplication of values and inclusion of placeholders, operations over it are more complex. For example, recursive query is necessary for traversing levels forming a hierarchy for the recursive relationship between members of the same type. This kind of queries are already included in the SQL:99 standard [130] and in commercial systems, for example, Oracle 10g [149]. Further, in this representation the hierarchy structure is not clear and it is necessary to retrieve all rows to reconstruct it, i.e., we must move to the data level to recover information about the

---

\(^8\)Symmetric hierarchies may also be represented using this structure, however this approach is used less frequently.
2.4. DATA WAREHOUSE HIERARCHIES

Figure 2.7: Transformation of the asymmetric hierarchy from Figure 2.6 b) to a symmetric one using placeholders.

schema. An additional disadvantage consists in using the same set of attributes for all hierarchy levels. Thus, the inclusion of additional attributes specific for a some level will force to include null values for members that do not belong to this level as can be seen in Figure 2.8 c).

Parent-child tables are mostly used when all hierarchy levels express the same semantics, i.e., when the characteristics of children and parents are similar (or the same), for example, when an employee has a supervisor that is also an employee. Also, the same measures are used for all members and in some situations, an aggregation of the measures of child members is not required; for example, the supervisor's salary does not need to be calculated aggregating salaries of his subordinates [135].

Currently, Microsoft Analysis Services proposes the implementation of asymmetric hierarchies using parent-child tables with several approaches for manipulating the measures:

1. Not allowing the inclusion of the measures in a fact table for the members that are not in the leaf level; here, aggregation to a parent level is done in a traditional way taking into account the measures of child members. The placeholders are used for the shorter branches.

2. Allowing the inclusion of measures for non-leaf members (the option members with data) with the following aggregation rules:

   (a) Aggregate to a parent level the measures of all its child members and its own measure.

   (b) Use in a parent member its own measures without aggregating those from its child members.

These options offer some flexibility in the management of asymmetric hierarchies. Nevertheless, regardless which option is chosen, the measures have to be the same for all hierarchy levels.
2.4.1.3 Generalized hierarchies

**Conceptual representation.** Sometimes a dimension includes subtypes that can be represented by a generalization/specialization relationship [4, 6, 115]. Moreover, the specialized subtypes can include their own hierarchy. An example is given in Figure 2.9. In this case the subtypes have a hierarchy with common levels (for example, Branch and Area) and specific levels (for example, Profession and Class for Person, and Type and Sector for Company).

In the ER representation the hierarchical paths cannot be clearly distinguished, in particular because higher hierarchy levels (for example, Area) can be included in a supertype without reference to the other hierarchy levels in its subtypes. Therefore, the information about levels forming a hierarchy and child-parent relationships existing between them cannot be retrieved. For example, in Figure 2.9 it is not clear that measures related to a customer that is a person can be aggregated using a hierarchy formed by Profession-Class-Branch-Area. Further, while both hierarchies can be represented independently repeating the common levels, the complexity of the schema reduces if it is possible to include shared levels characterized by the same granularity of aggregation. Also, to ensure adequate measure aggregations, the distinction between
2.4. DATA WAREHOUSE HIERARCHIES

Figure 2.9: ER representation of a dimension with different kinds of customers.

Specific and common hierarchy levels should be clearly represented in the model [80]. For representing such kinds of hierarchies we propose the graphical notation shown in Figure 2.10 where the common and specific hierarchy levels are clearly represented. We call such hierarchies generalized.

At the schema level a generalized hierarchy contains multiple exclusive paths sharing some levels as shown in Figure 2.10 a). All these paths represent one hierarchy and account for the same analysis criterion. At the instance level each member of the hierarchy only belongs to one path as can be seen in Figure 2.10 b). We propose to include the symbol ⊕ to indicate that for every member the paths are exclusive. Such a notation is equivalent to the xor annotation used in UML [23]. The levels at which the alternative paths split and join are called, respectively, splitting and joining levels.

In the example in Figure 2.10 the paths between both ⊕ symbols refer to attributes that can be used for aggregation purposes for the specialized subtypes from Figure 2.9. The lower path (i.e., containing the Profession and Class levels) corresponds to the Person subtype while the upper path (i.e., including the Type and Sector levels) indicates the Company subtype.

Not all levels that split must be joined. For example, Figure 2.11 shows a generalized hierarchy used for analyzing international publications at the university. Three kinds of publications are considered, i.e., journals, books, and conference proceedings. The latter can be aggregated to the conference level. However, there is not a common joining level for all paths.

Generalized hierarchies include a special case commonly referred to as non-covering hierarchies [135, 161]. The example given in Figure 2.12 represents a sales company having stores in different states. However, the geographical division in these states may vary, for example, skipping the division into counties for some of them. A non-covering hierarchy is a generalized hierarchy with the additional restrictions that at the schema level the root and the leaves are the same for all paths and the alternative paths are obtained by skipping one or several intermediate levels. At the instance level every child member has only one parent member, although the path length from the leaves

\footnotesize\textsuperscript{9}The dotted line in the figure (e.g., for branch 2) indicates the presence of members in the remaining levels
CHAPTER 2. CONVENTIONAL DATA WAREHOUSES

Figure 2.10: A generalized hierarchy for a sales company a) schema and b) examples of instances.

to the same parent level can be different for different members.

Notice that not all generalization/specialization hierarchies can be represented by generalized hierarchies in the multidimensional model. Recall that a generalization/specialization can be a) total or partial, and b) disjoint or overlapping [48]. Partial specializations induce an additional path in the generalized hierarchy relating the common levels. For example, if the generalization in Figure 2.9 were partial indicating that there are some customers that are considered neither person nor company, then an additional path between Customer and Branch is needed in the hierarchy of Figure 2.10 a).

On the other hand, for overlapping generalizations, different options can be given according to user requirements and availability of measures. Consider for example an overlapping specialization of Person in Student and Assistant. If measures are known only for the superclass Person, then only the hierarchy with common levels will be represented. If measures are known only for each subclass, i.e., for Student and Assistant, separate dimensions and fact relationships with corresponding measures can
2.4. DATA WAREHOUSE HIERARCHIES

be created for each specialization. However, significant programming effort is required for managing different dimensions with overlapping set of members and for developing aggregations procedures that allow to traverse common hierarchy levels. Another solution is to disallow these kinds of generalization/specialization for multidimensional hierarchies.

Logical representation. Several approaches were proposed to cope with the logical representation of generalized hierarchies: (1) representing heterogeneity within levels by creating separate tables for each hierarchy level of different paths [88]; (2) using a flat representation with null values for attributes that do not describe specific members [29, 110], for example, companies will have null values in attributes corresponding to persons; (3) using separate star schemas, i.e., separate fact tables and dimension tables for each path [11, 75, 101]; (4) creating one table for the common levels and another table for the specific attributes [11].

One disadvantage of most of these approaches is that common levels cannot be easily distinguished and managed. Further, the inclusion of null values requires specifying additional constraints to ensure correct queries (for example, not grouping Type with Profession in Figure 2.10). Even though these difficulties do not exist for the last solution (4), an additional level must be created in the table representing the common hierarchy levels for specifying the name of the table for the specific hierarchy levels. In consequence, queries managing metadata and data are required to access the tables of the specific levels, which is not an easy task to accomplish in SQL. Moreover, the traditional mapping of generalization/specialization to relational tables (e.g., [48]) gives also problems due to the inclusion of null values and the loss of the hierarchical structure.

Applying the mapping described in Section 2.3 to the generalized hierarchy given in Figure 2.10 yields the relations shown in Figure 2.13. Even though this mapping clearly represents the hierarchical structure, it does not allow to traverse only the common hierarchy levels. To ensure heterogeneity within levels as well as the possibility of
Figure 2.12: A non-covering hierarchy a) schema and b) examples of instances.

traversing either specific or common levels, we propose a more general solution using a non-traditional mapping to relational tables. An example of the relations for the hierarchy presented in Figure 2.10 is given in Figure 2.14. The mapping is as follows:

1. Create a separate relation for each level.

2. Include in every relation, except in the one that represents a splitting level (Customer in Figure 2.14) a foreign key that indicates the next hierarchy level.

3. Include for a splitting level two kinds of foreign keys: one that indicates the next specialized hierarchy level (e.g., Profession fkey in the relation Customer) and another that corresponds to a common level (e.g., Branch fkey in the relation Customer). Additionally, the splitting level could include an attribute that indicates the specific path, i.e., for a person or a company, to facilitate member grouping for aggregations.

The structure in Figure 2.14 allows to access different paths following the next links. When the splitting node is reached, the next link for specialized levels, for example,
2.4. DATA WAREHOUSE HIERARCHIES

37

for a person, can be chosen. When the generalized link is used, the hierarchy is the one that is common for all members, i.e., the levels between the splitting and joining levels are ignored allowing to analyze all customers together.

As for the snowflake structure, one disadvantage of this method is the necessity to apply join operations between several tables. However, an important advantage is the expansion of the analysis purposes taking into account specific and general characteristics of hierarchy members.

Generalized hierarchies do not satisfy the summarizability condition [111] because the mapping from the splitting level to the parent levels is incomplete; for example, not all customers roll-up to the Profession level. Thus, the aggregation mechanism should be modified when the splitting level is reached in a roll-up operation. Moreover, summarizability conditions can also vary for the joining level. To establish correct aggregation procedures, constraints proposed by Hurtado and Mendelzon [80] for heterogeneous hierarchies could be applied. On the other hand, the traditional approach can be used for aggregating measures for common hierarchy levels.

As a non-covering hierarchy is a special case of a generalized hierarchy, the solution for mapping to the relational model proposed for the latter can be also applied for the former. Nevertheless, since only some levels are usually skipped, in many applications (e.g., [135, 161]) such hierarchies are transformed by including placeholders in the missing intermediate levels, making the number of levels in each hierarchy path to be the same. In this way a non-covering hierarchy is converted into a symmetric hierarchy and the flat table or the snowflake structures can be used for its relational representation.

Currently it is not possible to manage generalized hierarchies in commercial tools. If the members differ in attributes and in hierarchy structure, the common solution is to treat the subtypes as separate dimensions, each one with their own hierarchy. In this way users are not able to analyze data according to common hierarchy levels and must cope with a more complex structure.

Figure 2.13: Relations for the generalized hierarchy from Figure 2.10.
The situation is different for non-covering hierarchies. For example, Microsoft Analysis Services allows the definition and manipulation of non-covering hierarchies, called *ragged hierarchies*. They can be represented in a flat table or in a parent-child table; the latter allows a ragged hierarchy to be also asymmetric\(^n\).

For a flat table representation, all levels must be used in a hierarchy definition (the path with the longest length is used). Thus, the members that do not include this level in a hierarchy will contain null values or placeholders. In order to obtain the desired display for these members, the dimension properties *Hide Member If* and *Visible* can be manipulated. In general, there are three possibilities for displaying non-covering hierarchies:

1. See all levels regardless of whether they have or not members. This is the default option (the property *Hide Member If* is set to *never hidden* and the property *Visible* is set to *true*) and the skipped levels are also presented with the null values or placeholders.

2. See only levels with members different from null values or placeholders (several options for the property of *Hide Member If* can be applied). In the example of Figure 2.12, the members of a County level will be displayed only for those members that are different from null or placeholders. However, Microsoft Analysis Services does not offer a correct management of this hierarchy. Even after installing the recommended service packs, the hierarchy can be displayed correctly in a Dimension Editor, yet cannot be manipulated in a Cube Browser: the paths that do not include some levels cannot be drilled-down after reaching a skipped level.

-----

\(^n\)Called unbalanced by Microsoft Analysis Services.
3. Ignore the additional levels. The property of Visible is set to false for the levels that are not included in all paths (County in Figure 2.12). This option requires additional manipulations of the property Member Keys Unique for the child members of a hidden level. Thus, in this option all members account for the same hierarchy structure.

The parent-child structure for manipulating non-covering hierarchies in Microsoft Analysis Services requires the inclusion of an additional column in a parent-child table that indicates the number of skipped levels. However, in this option the Microsoft Analysis Services does not offer adequate hierarchy representation and manipulation neither in the Dimension Editor nor in the Cube Browser. The installed service packs do not correct this problem either.

2.4.2 Non-strict hierarchies

Conceptual representation. For the simple hierarchies presented before we assumed that each link between child and parent levels has many-to-one cardinalities, i.e., a child member is related to at most one parent member and a parent member may be related to several child members. However, a many-to-many cardinality between child and parent levels is very common in real-life applications. For example, a diagnosis may belong to several diagnosis groups [161], a week may be part of two months, a banking account may belong to several customers [108], a mobile phone can be classified in different product categories, etc.

We call a hierarchy non-strict if at the schema level it has at least one many-to-many cardinality; it is called strict if all cardinalities are many-to-one. The fact that a hierarchy is strict or not is orthogonal to its type. Thus, the different kinds of hierarchies previously presented can be either strict or non-strict.

![Diagram of a non-strict hierarchy](image)

Figure 2.15: A symmetric non-strict hierarchy: a) schema and b) example of instances.
CHAPTER 2. CONVENTIONAL DATA WAREHOUSES

Figure 2.15 a) shows a symmetric non-strict hierarchy where a product can belong to different categories\(^\text{11}\). For example, a mobile phone may not only provide calling facilities but also may serve as a personal digital assistant (PDA) and as an MP3 player. In a strict hierarchy, this product can only belong to one category, i.e., phone. In a non-strict hierarchy it may be considered as belonging to three categories, i.e., phone, PDA, and MP3 players. Therefore, since at the instance level, a child member may have more than one parent member, the members form an acyclic graph (Figure 2.15 b)).

Notice the slight abuse of terminology. We use the term “non-strict hierarchy” for denoting acyclic classification graph [79]. We use this term for several reasons. Firstly, unlike the acyclic classification graph, the term hierarchy indicates that users require to see measures at different levels of detail. Secondly, the term hierarchy is already used by practitioners and, as we will see in the next section, several implementation solutions already exist for managing them. In particular SQL Server Analysis Services 2005 allows to include many-to-many child-parent relationships. Finally, the term hierarchy is also used by several researchers (e.g., [4, 115, 161, 199]).

Non-strict hierarchies cause the problem of double-counting measures during the roll-up operations when reaching the many-to-many cardinalities. Let us consider the example in Figure 2.16. It illustrates the sales of products with aggregations according to the Category and to the Department levels. Suppose that the product with sales equal to 100 is a mobile phone that also includes PDA and MP3 functionalities. In a strict hierarchy, this mobile phone can only belong to one category. Therefore, the sum of sales by categories and departments can be calculated without any problems as is shown in Figure 2.16 a). However, it might be interesting to also consider this mobile phone when calculating the sum of sales for PDA and MP3. Nevertheless, as can be seen in Figure 2.16 b), this approach causes an incorrect result since the phone's sales is counted three times instead of only once.

One of the possible solutions to the double-counting problem is to indicate how the measure is distributed between several parent members when reaching a many-to-many

\(^{11}\)This example is inspired from [79].
2.4. DATA WAREHOUSE HIERARCHIES

Employée
- Employee id
- Employee name
- Position

Section
- Section name
- Description
- Activity
- Responsible

Division
- Division name
- Type
- Responsible

Figure 2.17: A symmetric non-strict hierarchy with a distributing factor.

cardinality. For example, Figure 2.17 shows a symmetric non-strict hierarchy where employees may work in several sections. The model includes a measure that represents an overall employee's salary, i.e., a sum of salary paid in each section. It may be the case that the percentage of the time that an employee participate in each section is known. Therefore, in this case we propose to include an additional symbol \( \Theta \) called *distributing factor* that indicates how measures should be divided between several parent members when reaching many-to-many cardinalities in non-strict hierarchies.

Notice that for the mobile phone example the distribution of the sales measure between three categories according to the established percentages, e.g., 70% to phone, 20% to PDA, and 10% to MP3, not always may give meaningful information to users. In other situations, this distribution is impossible to specify. For example, suppose a user wants to analyze the daily account balances in a bank\(^\text{12}\). Some accounts are joint between several customers, however, there is not information how the account balance may be distributed between different customers of this joint account.

**Logical representation.** The traditional mapping of non-strict hierarchies as the one presented in Figure 2.15 a) to the relational model creates relations for representing levels and an additional relation for representing a many-to-many cardinality between levels. In current DW applications, the table representing the many-to-many cardinality, is called *bridge table* [101]. However, as we already explained for the example in Figure 2.16, this structure does not allow to aggregate measures correctly when a many-to-many cardinality is reached. On the other hand, different kinds of analysis that require some programming effort could be done. For example, users can analyze sales of the products that belong to more than one category.

Another mapping of non-strict hierarchies can be used when a measure distribution is not known or not needed for the analysis purposes. For example, Pedersen *et al.* [161] transform a non-strict hierarchy into a symmetric hierarchy by joining in one group the parent members participating in a many-to-many relationship. In our mobile phone example, a new category member will be created that represents the three categories together: mobile phone, PDA, and MP3 player. However, this solution introduces artificial categories and may not correspond to users' analysis requirements.

\(^{12}\) This example is inspired from [108].
Two solutions can be applied when a distributing factor is included in the model (Figure 2.17). The first one uses the traditional mapping of many-to-many cardinalities and includes an additional attribute that represents a distributing factor. Therefore, the above-mentioned bridge table will include not only references to members of child and parent levels, but also values required for the measure distribution (the table EmpSection in Figure 2.18). For example, for an employee X, the table EmpSection could include three rows: (1) employee X, section 1, 30%; (2) employee X, section 2, 50%; (3) employee X, section 3, 20%. Then, a special aggregation procedure must be implemented in order to aggregate measures correctly.

However, in some situations this solution may give approximate results. For example, suppose that the distributing factor represents the percentage of the time that an employee works in the specific section. If the employee has a higher position in one section, even though he works less time in this section, he may earn a higher salary. Therefore, applying the distributing factor to the measures representing the employee’s overall salary when reaching many-to-many cardinality may not give the exact result.

To solve this problem, another usual implementation solution transforms non-strict hierarchies into independent dimensions as shown in Figure 2.19 a)\(^\text{13}\). This solution corresponds to a different conceptual model shown in Figure 2.19 b) where the focus of analysis has been changed from employee’s salary to employee’s salary by section. Notice that this solution allows to represent the exact amount of salary paid without incurring to approximate results.

Nevertheless, using this solution even though the measure salary will aggregate correctly when applying the roll-up operation from the Section to the Division levels, the problem of double counting the same employee will occur \([161]\) as can be seen in Figure 2.20. In this example, five employees are considered. The table Employee facts contains the assignment of employees to sections. The count of the number of employees that work in each section gives correct results. However, the aggregated values for each section cannot be reused for calculating the number of employees for every division since some employee (E1 and E2 in Figure 2.20) will be counted twice.

Notice that non-strict hierarchies without distributing factor (Figure 2.15 a)) cannot

\(^{13}\)For simplicity, we omit other dimensions, such as the Time dimension.
2.4. DATA WAREHOUSE HIERARCHIES

![Diagram](image)

Figure 2.19: Transformation of a non-strict hierarchy into a strict hierarchy. a) relational tables and b) the MultiDimER representation.

![Table](image)

Figure 2.20: Double counting problem for non-strict hierarchies.

be represented as separate dimensions as in Figure 2.19 b) since the problem of double-counting of measures will occur when grouping over the dimension that was representing the parent level in the many-to-many cardinality, e.g., Category in Figure 2.15 a).

Therefore, both options for implementing non-strict hierarchies with the distributing factor require programming effort for correct management during aggregation operations. They offer the same possibilities of analysis. However, the fact relation in Figure 2.19 a) contains more detailed data than that of Figure 2.18. In the former, there is one row per each section in which an employee works; in the latter there is one row per employee. Thus, if an employee works in many sections, the fact table in Figure 2.19 a) will include more rows. Table 2.1 includes some aspects that can be considered when choosing one of the proposed implementations.

Even though non-strict hierarchies exist in many real-world situations, no much research is dedicated to them. The search for solutions that allow better management of these hierarchies is considered of fundamental importance [79]. In this section we only analyzed the case of non-strict symmetric hierarchies. When the other hierarchy types presented before are non-strict, a combination of the proposed mapping for the specific hierarchy and the mapping for a non-strict hierarchy should be applied.
Table 2.1: Comparison of implementations of non-strict hierarchies using a bridge table and separate tables.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Bridge table (Figure 2.18)</th>
<th>Separate tables (Figure 2.19)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table size</td>
<td>The fact table is smaller.</td>
<td>The fact table is bigger if child members are related to many parent members. The additional foreign key in the fact table will also increase space.</td>
</tr>
<tr>
<td>Additional structure</td>
<td>Additional information about child-parent relationship and distribution factor (if exists) must be stored separately.</td>
<td>Measures are only included in the fact table.</td>
</tr>
<tr>
<td>Performance</td>
<td>Additional join operations, calculations, and programming effort are needed to aggregate measures correctly.</td>
<td>Measures in the fact table are ready for aggregation across the hierarchy.</td>
</tr>
<tr>
<td>Type of application</td>
<td>Useful for applications having only few non-strict hierarchies and other strict hierarchies referring to the same measures.</td>
<td>Its implementation should be evaluated taking into account a space-time trade-off.</td>
</tr>
<tr>
<td>Changes in time</td>
<td>Useful when the information about measure distribution does not change in time.</td>
<td>Can easily represent changes in measure distribution that occur in time.</td>
</tr>
<tr>
<td>Type of application</td>
<td>Useful for applications having only few non-strict hierarchies and other strict hierarchies referring to the same measures.</td>
<td>Its implementation should be evaluated taking into account a space-time trade-off.</td>
</tr>
<tr>
<td>Current OLAP implementation</td>
<td>Current tools manage to some extent this type of hierarchy.</td>
<td>Current tools can manage it. In some cases the problems of double-counting may occur.</td>
</tr>
</tbody>
</table>

2.4.3 Multiple alternative hierarchies

Conceptual representation. Multiple alternative hierarchies represent the situation where at the schema level there are several non-exclusive simple hierarchies sharing some levels. However, all these hierarchies account for the same analysis criterion. At the instance level such hierarchies form a graph since a child member can be associated with more than one parent member belonging to different levels. In such hierarchies it is not semantically correct to simultaneously traverse the different composing hierarchies. The user must choose one of the alternative hierarchies for analysis. An example is given in Figure 2.21 a) where the Time dimension includes two hierarchies corresponding to different calendar subdivisions: Day-Month-Quarter-Year and Day-Week-Year. As can be seen in the example it is meaningless to combine both hierarchies. The combination means to use both hierarchies simultaneously. In the example this would
2.4. DATA WAREHOUSE HIERARCHIES

give intersections, such as Quarter-1-2006 and Week-7-1995, that can be confusing for users. Hierarchies are modeled as multiple alternative when the user requires to analyze measures from the same perspectives (e.g., time) using different alternatives for aggregations.

Figure 2.21: A multiple alternative hierarchy: a) conceptual schema and b) relations.

Notice the difference between generalized and multiple hierarchies (Figures 2.10 and 2.21). Although both hierarchies share some levels and use only one analysis criterion, they represent different situations. In a generalized hierarchy a child member is related to one of the paths, whereas in multiple hierarchies a child member is related to all paths, and the user must choose one of them for analysis.

Logical representation. For multiple alternative hierarchies, the traditional mapping to relational tables can be applied (Figure 2.21 b)). Since the measures from the fact relation will totally participate in each composing hierarchy, measure aggregation can be performed as for simple hierarchies. Aggregated measures for a common hierarchy level can be reused while traversing one of the hierarchies.

Notice that even though generalized and multiple hierarchies can be easily distinguished at the conceptual level (Figures 2.10 a) and 2.21 a)), this distinction cannot be made at the logical level (Figures 2.13 and 2.21 b))

Multiple alternative hierarchies can be implemented in Microsoft Analysis Services with the goal of improving system performance and storage requirements. This is realized by including a single key in a fact table and sharing the aggregate values of common levels (for example, Year in Figure 2.21). These hierarchies must include a
naming schema that indicates the presence of more than one hierarchy. A name includes two parts separated by a dot: the first part is common for all composing hierarchies and the second part is a unique hierarchy name. In the example of Figure 2.21 a), we can create two hierarchies called Time.MonthQuarter and Time.Week. However, the tool does not allow the adequate manipulation of multiple alternative hierarchies. For example, both hierarchies from Figure 2.21 a) can be combined giving meaningless intersections such as Quarter-2-2003 and Week-5-1998. Even though the measures are manipulated adequately giving an empty display for these meaningless intersections, a better solution should be provided. The tool could allow to automatically switch between the composing hierarchies. For example, if the user is analyzing measures using the hierarchy Time.MonthQuarter and requests (click-and-drag) another hierarchy (Time.Week), the first hierarchy should be excluded from current analysis.

2.4.4 Parallel hierarchies

Parallel hierarchies arise when a dimension has associated several hierarchies accounting for different analysis criteria. Such hierarchies can be independent or dependent.

2.4.4.1 Parallel independent hierarchies

Conceptual and logical representations. In a parallel independent hierarchy, the different hierarchies do not share levels, i.e., they represent non-overlapping sets of hierarchies. Parallel independent hierarchies may be composed of different kinds of hierarchies. Figure 2.22 shows an example of a dimension having both a symmetric and a non-covering hierarchy used for different analysis criteria. The hierarchy Type is used for grouping products according to different categories or departments while the hierarchy Distributor location groups them according to different divisions or regions of distributors of these products.

![Parallel independent hierarchy composed of a symmetric and a non-covering hierarchies.](image)

As parallel independent hierarchies are a combination of the hierarchies previously presented, their logical mapping consists in combining the mappings from the specific hierarchy types. Current OLAP tools allow to manage this kind of hierarchies.
2.4.4.2 Parallel dependent hierarchies

Conceptual representation. A parallel dependent hierarchy has different hierarchies sharing some levels. The example given in Figure 2.23) represents an international company that requires sales analysis for stores located in different countries. The Store dimension contains two hierarchies: one representing the geographic division of the store address and the other one representing the organizational division of the company. The first hierarchy is non-covering and includes the levels Store, City, County (for the States that have it), State, and Country. The other one is a symmetric hierarchy and includes the levels Store, Sales district, State, and Sales region. Both hierarchies share the common levels of State playing different roles. In order to ensure an unambiguous distinction of the level shared between different hierarchies, the symbol of the analysis criterion is also included.

Logical representation. The result of the traditional mapping to the relational model gives the relations presented in Figure 2.24. Notice that both multiple and parallel dependent hierarchies share some levels. These two types of hierarchies can be easily distinguished at the conceptual level (Figures 2.21 a) and 2.23). However, the logical level representation (Figures 2.21 b) and 2.24) looks similar in spite of several characteristics that differentiate them:
Parallel dependent hierarchies account for different analysis criteria; thus, the user can use them independently during the analysis process performing combinations between levels of different hierarchies without producing meaningless intersections, for example, "which are the sales values for the store in the city A that belongs to sales district X".

It is not always possible in parallel dependent hierarchies to reuse aggregated measures for shared levels. For instance, if instead of the Store level we include the Sales employee level keeping the same hierarchies, different results will be obtained aggregating measures traversing one or another hierarchy since sales employees can live in one state and work in another.

Since members of shared levels can participate in parallel dependent hierarchies playing different roles, it is necessary to create views to avoid unnecessary data repetition.

Parallel dependent hierarchies can be implemented in Microsoft Analysis Services treating them as separate hierarchies with a repetition of common levels.

2.4.5 Hierarchy categorization

To give a more general view of our proposal for different kinds of hierarchies and relationships existing between them, we present the classification of hierarchies us-
2.5. Example

The example in Figure 2.26 includes all hierarchy types described in previous sections except the asymmetric hierarchy. This example shows three fact relationships indicating that the focus of analysis is related to sales, employees' salaries, and incentive sales programs.

As shown in the figure, the MultiDimER model allows to represent shared levels as well as shared dimensions. Sharing levels allows to reuse existing data while sharing dimensions opens the possibility to analyze measures presented in different fact relationships\(^{14}\). For example, the level Month is shared between two hierarchies (the Sales time and the Employee time) while the level State is shared between three hierarchies.

\(^{14}\)This operation is called *drill across* \([101]\) and requires common dimensions between two or more fact relationships.
CHAPTER 2. CONVENTIONAL DATA WAREHOUSES

Figure 2.26: Example of schema containing several hierarchies reusing hierarchy levels.
(the Sales organization, the Store location, and the Customer location). The Store dimension with two hierarchies Sales organization and Store location is shared between Sales and Employees' salaries fact relationships. Thus, the measure Sales (from the Sales fact relationship) can be analyzed together with the Amount of salary paid (from the Employees' salaries fact relationship) for different stores in different periods of time.

Further, the MultiDimER model allows to define several child-parent links between the same levels, for example, the links between the Employee-Section relationship for the Works and Affiliated hierarchies. Each link corresponds to a different analysis criterion and has an associated set of instances.

The relational mapping of this example can be obtained by applying the mapping of the different hierarchies as explained in previous sections.

### 2.6 Related work

The necessity of a conceptual modeling phase for database design has been acknowledged for several decades and has been studied in many works (e.g., [23, 33]). However, the analysis presented in [180] shows the little interest of the research community in conceptual multidimensional modeling. Further, the proposed models do not cope with the different kinds of hierarchies existing in real-world applications.

The works reviewed next fall in three categories. First, works describing different types of hierarchies. Secondly, works referring to logical representations of hierarchies. Finally, we briefly cover the commercial solutions for implementing hierarchies. For a better comparison, in the description that follows we use our terminology for the different kinds of hierarchies.

There are several proposals for DW/OLAP conceptual models that usually refer to general concepts, such as dimensions, measures, and rarely consider different kinds of hierarchies. These models are based on either UML (e.g., [4, 115]) or the ER model (e.g., [183, 199]). Other works (e.g., [61, 82, 174, 201]) include specific graphical notations for expressing the semantics of the multidimensional model. Further, there are works (e.g., [80, 161]) that formally define some kinds of hierarchies described in this chapter, however, they mainly discuss the summarizability conditions and provide some solutions for correct measure aggregations in presence of the so-called heterogeneous hierarchies [80].

Table 2.2 compares multidimensional models that, at the best of our knowledge, cope with hierarchies. We use three symbols for their comparison: — when no reference to the hierarchy exists, ± when only a description and/or definition of the hierarchy is presented, and √ when a description and a graphical representation are given. If a different name for a hierarchy is proposed, it is included in the table. All models include explicitly or implicitly the strict and parallel independent hierarchies. Further, none of the models takes into account different analysis criteria applied for hierarchies; in consequence, the multiple alternative and parallel dependent hierarchies cannot be distinguished.

Pedersen et al. [161] describe different types of hierarchies appearing in a healthcare application. As can be seen in Table 2.2 they distinguish several kinds of hier-...
Table 2.2: Comparison of the conceptual models including different kinds of hierarchies.

<table>
<thead>
<tr>
<th>Hierarchy / Model</th>
<th>Symmetric</th>
<th>Asymmetric</th>
<th>Generalized</th>
<th>Non-Covering</th>
<th>Non-strict</th>
<th>Multiple Alternative or Parallel dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>[161]</td>
<td>Onto</td>
<td>Non-onto</td>
<td></td>
<td>Non-covering</td>
<td>Non-strict</td>
<td>Multiple</td>
</tr>
<tr>
<td>[62]</td>
<td>Simple</td>
<td>—</td>
<td>Multiple</td>
<td>±</td>
<td>—</td>
<td>Multiple Alternative</td>
</tr>
<tr>
<td>[199]</td>
<td>✓</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>Non-strict</td>
<td>Multiple</td>
</tr>
<tr>
<td>[80]</td>
<td>Strictly Homogenous</td>
<td>Homogenous</td>
<td>Heterogeneous</td>
<td>Heterogeneous</td>
<td>—</td>
<td>Heterogeneous</td>
</tr>
<tr>
<td>[168]</td>
<td>Classification</td>
<td>Aggregation</td>
<td>Multiple Aggregation</td>
<td>±</td>
<td>—</td>
<td>Multiple Multiplicity</td>
</tr>
<tr>
<td>[4]</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>±</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>[115]</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>✓</td>
<td>✓</td>
<td>Multiple, Alternative</td>
</tr>
<tr>
<td>[201]</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>✓</td>
<td>±</td>
<td>—</td>
</tr>
<tr>
<td>[61, 183, 197]</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>[143]</td>
<td>Level Based</td>
<td>Value Based</td>
<td>Value Based</td>
<td>Value Based</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

archies presented in this thesis. However, as in other models, multiple alternative or parallel dependent hierarchies cannot be distinguished and generalized hierarchies are not considered.

Hissemann et al. [82] adopt a conceptual model based on functional dependencies and a multidimensional normal form [110] to ensure correctness of schemas. Their model describes two kinds of hierarchies: symmetric and what they called multiple. The latter category is further divided into optional and alternative hierarchies. They implicitly refer to non-covering hierarchies proposing to insert the value “other” in missing levels.

Tryfona et al. [199] propose the StarER model, an extension of the ER model. They refer to different kinds of hierarchies based on Pedersen et al. [161] work and show that their graphical model is able to represent different types of hierarchies as can be seen in Table 2.2.

Hurtado and Mendelzon [80] define homogenous and heterogeneous hierarchies as indicated in the table. The former include symmetric and asymmetric hierarchies. The latter can be used for generalized, non-covering, and multiple alternative or parallel dependent hierarchies. They extend the notion of summarizability to heterogeneous hierarchies by including constraints that can be applied for the implementation of the aggregation procedures for different kinds of hierarchies proposed in this work.

Purabbas and Rafanelli [168] define a classification hierarchy that corresponds to
2.6. RELATED WORK

symmetric hierarchies. Their definition of aggregation hierarchies includes asymmetric, generalized, and non-covering hierarchies without making clear distinction between them. They do not allow the existence of non-strict hierarchies. Moreover, their definition of multiple hierarchies can be applied to generalized, multiple alternative or parallel dependent hierarchies.

Bauer et al. [11] describe the problem of using hierarchies that include common levels and attributes specific to different groups of the members. To handle adequately this kind of hierarchies, they give two solutions using a relational and an object-relational approach. Even though their relational implementation provides an interesting approach, the access to levels of sub-tables requires queries that manage metadata and data that is difficult to accomplish in current database systems.

Abelló et al. [4] propose a conceptual multidimensional model called YAM2 based on UML. Their model includes symmetric, multiple alternative, and non-strict hierarchies. Further, for asymmetric and non-covering hierarchies, they propose a logical-level solution to include "dummy values" for making them symmetric and covering.

Luján-Mora et al. [115] represent symmetric, non-covering, non-strict, and multiple alternative or parallel dependent hierarchies using the UML notation. Further, they distinguish a categorizing dimension based on the specialization/generalization relationship. Using this representation, they do not focus on hierarchies that include common and specific levels as proposed in this work for generalized hierarchies.

Tsois et al. [201] propose the MAC model for representing OLAP scenarios. Even if they refer informally to symmetric, non-covering, and non-strict hierarchies, in their graphical representation it is difficult to distinguish the different kinds of hierarchies.

There are other conceptual models that include main features for multidimensional modeling, such as dimensions, facts, and measures [61, 183, 197]. However, these models do not focus on the different kinds of hierarchies, thus they only represent usual hierarchies as can be seen in Table 2.2.

The Object Management Group (OMG) proposes the Common Warehouse Model (CWM) [143] as a standard for representing DW and OLAP systems. This model provides a framework for representing metadata about data sources, data targets, transformations and analysis, processes and operations for creation and management of warehouse data. This model is represented as a layered structure consisting of a number of submodels. One of them, the resource layer, defines which models can be used for representing DW data and includes the relational model as one of them. Further, the analysis layer presents a metamodel for OLAP, which includes a dimension and a hierarchy. Hierarchies can be level or value based. The former correspond to symmetric hierarchies, the latter represent hierarchies where the members may be classified according to their distance from a common root level (for example, from Bank level in Figure 2.6), i.e., they may include asymmetric, generalized, and non-covering hierarchies. Since a level can participate in several child-parent relationships but these relationships and levels cannot belong to more than one hierarchies, multiple hierarchies are implicitly considered in the CWM standard. Further, a dimension may have several hierarchies, thus parallel independent hierarchies may also be seen as a part of the CWM specification.

Current commercial OLAP tools do not allow conceptual modeling of hierarchies. They usually provide a logical-level representation limited to star or snowflake schemas.
However, there are some exceptions, such as ADAPT [27] modeling tool for OLAP. It introduces new features for multidimensional conceptual modeling (for example, dimension scope, dimension context) with the limitation of allowing the graphical representation for only commonly-used hierarchies, i.e., symmetric and strict.

The logical schema for DWs is usually represented using relational tables mainly based on the star or snowflake structures. In the previous sections we already referred in details to works that use the relational approach for representing the different kinds of hierarchies, where sometimes complex hierarchies are transformed into simpler ones. For example, we discussed the approaches taken by Pedersen et al. [161] for transforming asymmetric, non-covering, and non-strict hierarchies, or the solutions proposed by Kimball et al. [101] for managing symmetric, generalized, and non-strict hierarchies. We also discussed the advantages and the disadvantages of the proposed solutions. Some of these solutions (e.g., [20, 88]), even though ensure a correct mapping that captures the semantics of the hierarchy, they produce a significant amount of relations.

Commercial products, such as Microsoft Analysis Services, Oracle, Informix MetaCube can cope with some hierarchies presented in this chapter. For example, Microsoft Analysis Services allows to manage symmetric, asymmetric, non-covering, parallel, and to some extent multiple hierarchies as was described in the corresponding sections above. Even though this tool includes programming capabilities through MDX (Multidimensional Expressions), ADO/MD (ActiveX Data Objects/Multidimensional), DSO (Decision Support Objects), and options such as Custom Rollup, a hierarchy structure must be specified as a star, a snowflake, or a parent-child schema, and as explained before, this has some limitations. The logical level in Microsoft Analysis Services is based on either the ROLAP technology using by default SQL Server 2000 for storing the tables or the MOLAP storage located in the Analysis Server. Also, a combination of both kinds of storage may be used according to user performance requirements and disk space.

Oracle [147] distinguishes level-based dimensions to implement symmetric hierarchies, and parent-child structures to represent asymmetric hierarchies. A dimension can have associated several hierarchies and the same levels can be used in different hierarchies, thus allowing parallel dependent hierarchies. The non-strict and multiple hierarchies presented in this work are not allowed. Oracle also allows some variations in implementing hierarchies, such as null values in levels, different lowest levels, and values mapped to different leaf levels (similar to the option "member with data" already explained for Microsoft Analysis Services). Oracle offers the integration of both relational and multidimensional technologies (using the so-called Analytic Workspace), to facilitate access to both kinds of data through SQL and PL/SQL statements, to improve performance, and to extend functionality for specific OLAP queries. For both kinds of storage, dimensions, measures, hierarchies, levels, and attributes should be defined.

The limitations of other OLAP tools, such as Informix MetaCube, are already described in, for example, [75, 115].
2.7 Summary

Data warehouse (DW) and on-line analytical processing (OLAP) systems support the decision-making process giving access and manipulation facilities for historical data. DWs are defined using a multidimensional view of data including dimensions, hierarchies, and measures. OLAP systems allow interactively querying DW data using operations such as drill-down and roll-up, which need hierarchies in order to automatically aggregate the measures for analysis purposes.

A hierarchy represents some organizational, geographic, or other type of structure that is important for analysis. However, although many kinds of hierarchies can be found in real-world applications, current OLAP systems manage only a limited number of them. Thus, when the user requires complex multidimensional analysis including several kinds of hierarchies, they are difficult to model. Further, since there is a lack of the logical level representation for different kinds of hierarchies, designers must apply different “tricks” at the implementation level to transform some hierarchy types into simple ones.

We considered that it is necessary to be able to represent the different elements of a multidimensional model including the different types of hierarchies at a conceptual as well as at a logical level. We proposed the MultiDimER model, which includes graphical notations for different elements of a multidimensional model and we provided the mapping to the relational model.

We classified hierarchies taking into account their differences at the schema as well as at the instance levels. We distinguish simple and multiple hierarchies. The latter are composed of one or more simple hierarchies accounting for the same analysis criterion. Moreover, simple hierarchies include further types: symmetric, asymmetric, and generalized hierarchies. Also, non-covering hierarchies are treated as a special case of generalized hierarchies. For each of these simple hierarchies, another specialization can be applied, making them strict or non-strict. The latter allows many-to-many cardinality between child and parent levels defining graphs at the instance level. Finally, we distinguish the situation where several hierarchies accounting for different analysis criteria may be attached to the same dimension. Depending on whether they share or not common levels, we called them parallel dependent and parallel independent hierarchies, respectively.

For the mapping to the relational model, we used a traditional approach since the MultiDimER model is based on the ER model. We proposed some modifications for generalized and non-strict hierarchies. We also discussed current approaches for some types of hierarchies comparing them and giving the indications in which situations different mappings work better. Further, we described which of these hierarchies can be currently used in commercial products using as an example Microsoft Analysis Services from SQL Server 2000.

The hierarchy representation in the MultiDimER model uses notations that allow a clear distinction of each type of hierarchy. Most of the existing conceptual multidimensional models do not distinguish between these kinds of hierarchies. Further, even though the mapping to logical level is based on well-known rules, it does not completely represent the semantics of each hierarchy: generalized, multiple, and parallel dependent hierarchies cannot be distinguished in the relational model. Therefore, using the
MultiDimER model we are able to capture in a better way meaning of data, expressing with more clarity the semantics particular to DW and OLAP systems. In this way, we gave to users, designers, and developers the possibilities to express, design, and implement different kinds of hierarchies.
Chapter 3

A conceptual model for spatial data warehouses

It is estimated that about 80% of data stored in databases has a spatial or location component [179], therefore, the location dimension has been widely integrated in DW and OLAP systems. However, this dimension is usually represented in an alphanumeric, non-cartographic manner (i.e., using solely the place name) since these systems are neither able to store nor to manipulate spatial data. Nevertheless, it is well known that the inclusion of spatial data in the analysis process can help to reveal patterns that are difficult to find otherwise.

Taking into account the growing demand for incorporating spatial data in the decision-making process, we extend a multidimensional model by the inclusion of spatial data. We realize such extension defining a conceptual model and proposing its subsequent logical representation.

We define a spatial level, which may have spatial attributes. We also refer to the definition of spatial hierarchies extending by means of examples the previous hierarchy classification (Section 2.4) by inclusion of spatial levels. Additionally, we present the classification of the topological relationships existing between spatial levels forming hierarchies according to the required procedures for measure aggregations. We include various scenarios when more than one spatial dimension is represented in a multidimensional model. Since these dimensions may be topologically related, we introduce a new concept of spatial fact relationships. Further, we extend the analysis to spatial measures allowing them to be represented by either a geometry or a numeric value calculated using spatial or topological operators. We also discuss the necessity of having spatial aggregation functions for measures when hierarchies are presented in the model.

This chapter is organized as follows. Section 3.1 introduces concepts related to spatial databases. Section 3.2 presents our rationale for choosing logical representation based on a spatially-extended object-relational model. Section 3.3 include the definition of the spatially-extended MultiDimER model. Sections 3.4, 3.5, and 3.6 include specific aspects for conceptual and logical representations of spatial levels, spatial hierarchies, and spatial fact relationships with (spatial) measures, respectively. Section 3.7 presents the metamodel of the spatially-extended MultiDimER model. Finally, related work are surveyed in Section 3.8 and Section 3.9 presents summary of this chapter.
3.1 Spatial databases: General concepts

Conceptual representation of spatial data Spatial databases have been used for several decades for storing and managing spatial data. They allow to include the description of the spatial properties of the real-world phenomena. There are two complementary ways of modeling spatial data for spatial database applications: object-based and field-based data modeling. The former decomposes space into identifiable objects describing their shape, for example, representing a road as a line or a state as a surface. The latter is used to represent phenomena that vary over space, associating to each point in a relevant space extent the value that characterizes the feature at that point, such as temperature, altitude, soil cover, pollution.

In this thesis, we refer to the object-based spatial data modeling. The possible extension for the inclusion of the field-based spatial data is described in Future work (Section 6.2.2). Next, we present in more details concepts related to spatial objects, which are useful for a better understanding of our proposal for spatial DWs.

Spatial objects A spatial object corresponds to a real-world entity for which the application needs to keep its spatial characteristics, for example, its shape and location. Spatial objects consist of a thematic (or description) component and a spatial component. The thematic component is represented using traditional DBMS data types, such as integer, string, and date; it contains general characteristics of the spatial object. For example, a state object may be described by its name, population, area, and major activity. The spatial component includes geometry, which can be point, line, surface, or a collection of these types. This geometry is represented and managed using two-dimensional space. The possible extension for the inclusion of spatial objects in three-dimensional space is described in Future work (Section 6.2.2).

A point is a so-called 0-dimensional geometry allowing to represent an object for which only its location in space, but not its extent is relevant. For example, the city can be modeled as a point when it is included in a model describing the state. A point requires two coordinates for its representation in two-dimensional space. A line is a so-called 1-dimensional geometry usually used for modeling connections in space [69], such as roads, rivers, phone cables. It is represented as a set of connected points. A surface is a 2-dimensional geometry used for representing an object having an extent in space, such as a state or a lake. It is usually represented as a set of connected points that delimits the boundary of the spatial object.

Reference systems The locations of points are given with respect to some coordinates of the plane, i.e., spatial reference system. The latter represents a function that allows to associate real locations in space to geometries of coordinates defined in mathematical space and vice versa [146]. For example, projected coordinate systems give Cartesian coordinates that result from mapping a point on the Earth's surface to a plane.

Topological relationships Spatial objects can relate to each other forming topological relationships, i.e., relationships that are invariant when the spatial objects are manipulated (for example, translated, rotated, scaled, etc.). Egenhofer [44] defined a
set of eight different binary topological relationships for surfaces. His definition is based on representation of a spatial extent by a set of points, composed of three subsets: the boundary, which delimits the extent, the interior of the extent, i.e., the points within the boundary, and the exterior of the extent, i.e., the points outside the boundary. His proposal is based on the intersection of interior, boundary, and exterior of two participating objects giving the so-called nine-intersection matrix \([46]\). Different topological relationships are defined, such as intersects, meets, contains. For example, A meets B means that the boundaries of A and B share some common point(s), their interiors are disjoint, and their exteriors have also common points.

The topological relationships on other pairs of spatial data types, for example, points and surfaces or points and lines can be defined in a similar manner. Several proposals for this extension already exist (e.g., \([14, 126]\)). Further, the SQL/MM standard \([87]\) uses the nine-intersection matrix allowing to use combinations of different spatial data types. It also defines boundary, interior, and exterior for every spatial data type included in the standard.

The usual practice in modeling spatial objects and topological relationships between them is to use the pictograms. For example, the conceptual spatio-temporal model MADS \([160]\) uses the pictograms shown in Figure 3.1.

\[\text{Figure 3.1: Pictograms for: a) spatial data types and b) topological relationships.}\]

**Implementation models** Object- and field-based data models represent spatial data at the abstract level. At the implementation level the so-called raster and vector data models are used.

*Raster* data model is structured as an array of cells (referred to as pixels), where each cell represents the value of an attribute for a real location \([48]\). This cell is addressed by its position in the array (row and column number). Raster can be used for representing different types of spatial objects, for example, a point may be represented as a single cell, a line by a sequence of neighboring cells, and a surface by a collection of contiguous cells.

In *vector* data model objects are constructed from points and lines as primitives \([178]\). A point is represented by its pair of coordinates whereas more complex linear and surfacic objects used structures (lists, sets, arrays) based on the point representation. Vector data model is more adequate for representing spatial data with clear boundaries in opposite to raster model where storage can be extremely inefficient for a large uniform area with no special characteristics \([214]\). For example, representing a city in a raster will include a set of cells that covers city's interior having city's name as an attribute value for each cell.
Models for storing collections of spatial objects

To store a collection of spatial objects using vector representation, spaghetti, network, and topological models can be used [178]. By considering collections, we focus not only on the individual objects but also on the relationships existing between them.

In the spaghetti model, the geometry of any spatial object of the collection is described independently of other objects. No topology is stored in such a model and it must be computed "on the fly" if required for analysis purposes. This structure introduces redundancy in the representation. For example, if two surfaces representing states share a common boundary line, this line will be defined twice, once for every surface. Therefore, a risk of inconsistency is incurred if, for example, the same boundary have slightly different coordinates due to different sources of information [178]. On the other hand, this approach is simple since it facilitates insertions of new objects, i.e., users do not need to know a priori the topological relationships between different spatial objects. In spite of the redundant representation, in many applications it requires less volumes of data compared to the network or topological models.

The network and topological models describe geometries representing the existed topology between lines and surfaces, respectively. These models define a set of different geometry types, such as nodes, arcs, and use them to represent a collection of spatially-related objects. Using the previous example, for defining a common boundary between two surfaces, first the definition of the arc representing boundary is introduced. Then, the definition of each surface refers to this arc. Therefore, using this representation queries for topological relationships are very efficient. For example, to retrieve states having a common boundary, the search for the surfaces referencing the specific arc is realized. However, the network and topological models are more complex and they have less flexibility for introducing new spatial objects [56] since users must know a priori the topological relationships existing between spatial objects.

Architecture

Regardless of the chosen structure, i.e., raster or vector, spaghetti or topological, to store spatial data two different architectures may be used: dual and integrated [204]. The former is based on separate management systems for spatial and non-spatial data while the latter extends DBMSs with spatial data types (for example, point, line, and surface) and functions (for example, overlap, distance, and area) [204].

A dual architecture is used by traditional Geographic Information Systems (GISs). It requires heterogeneous data models to represent spatial and non-spatial data. Spatial data is usually represented using proprietary data structures [204], which implies difficulties in modeling, use, and integration, i.e., it leads to increasing complexity of system management.

On the other hand, spatially-extended DBMSs provide support for storage, retrieval, query, and updating of spatial objects while preserving other DBMS functionalities, such as recovery techniques and optimization [178]. These spatially-extended DBMSs allow users to define an attribute of a table as being of spatial type, to retrieve topological relationships between spatial objects using spatial operators, to speed up spatial queries using spatial indexes, and so on. It also facilitates spatially enabling applications since spatial data is stored together with non-spatial data.

Several commercial DBMSs support the management of spatial data, for example, Oracle Spatial, Spatial Extender of IBM's DB/2, or PostGIS of PostgreSQL. This sup-
3.2 Motivation

In this section, we present our rationale for transforming a multidimensional model into a logical model based on an object-relational representation. We also refer to advantages of using an integrated architecture available in spatially-extended DBMSs (for example, Oracle Spatial). Finally, we refer to the importance of preserving semantics during the transformation from conceptual into logical schemas.

3.2.1 Using the object-relational model

In general, logical-level implementations for spatial DBs can be based on relational, object-oriented, or object-relational approaches [208]. The relational model has well-known limitations for representing complex, non-atomic data. For example, since spatial features are modeled using only conventional atomic attribute types, a boundary of a surface are stored as a set of rows each having the coordinates of two points representing a line segment forming the boundary [51, 186]. Therefore, the relational model imposes on users the responsibility of knowing and maintaining the groupings of rows representing the same real-world object in all their interactions with the database [36].

The object-oriented approach can solve these problems using complex objects for representing geometry. However, a pure object-oriented approach has not been yet widely integrated into current DBMSs. On the other hand, the object-relational model preserves the foundations of the relational model while extending its modeling power organizing data using an object model [107]. The object-relational model allows attributes to have complex types, i.e., inherently groups related facts into a single row [36]; for example, a surface’s boundary can be represented in one row. In addition, object-relational features are also included in the SQL:2003 standard [130, 131] and in leading DBMSs, such as Oracle or Informix.

As an example of object-relational DBMSs we use Oracle 10g. Oracle allows to define object types structured by users according to requirements for representing real-world objects. For example, an object type describing an address can contain several attributes composing this address, such as street, number, city, province. These object types can be used as a data type of a column in relational tables, as a data type of the attributes in other object types allowing nested structures, or as a type for a table. The latter is called an object table. It is similar to a conventional relational table with the difference that column types of this table correspond to the attributes of the object type used for the table declaration. Object tables include additionally an object-identifier column representing surrogates. Object-identifiers can be based on primary
keys or automatically generated by the DBMS. They can be used for establishing links between tables using a special ref type. A ref type is always associated with a specified object type. For example, in the table representing Client data, the column for the address can refer to the address object type.

3.2.2 Using spatial extensions of DBMSs

Current DBMSs with spatial extensions, such as Oracle, Informix, and Ingres are sufficient for storage, retrieval, and simple analysis of spatial data [203]. Even though these systems may provide raster and vector data representations as described in Section 3.1, we choose the latter since it is inherently more efficient in its use of computer storage than raster, because only points of interest need to be stored.

Oracle 10g Spatial allows to represent geometries under the object-relational model. The basic geometric types provided by Oracle are point, line string, and polygon. Line strings can be straight or curved shapes. Polygons have edges that may be straight or curved shape. Circles and rectangles are particular kind of polygons. Compound line strings and polygons are made of a mix of straight-line and circular-arc segments.

Oracle spatial model is organized hierarchically in layers, composed of geometries, at their turns composed of elements. Elements are the basic components of geometries. Elements can be of any basic geometric types. A geometry consists either of a single element or of an ordered list of elements. Complex geometries are modeled using a list of elements, for example, a set of islands is modeled by an ordered list of polygons, and islands within a lake is represented by a list of two polygons (exterior and interior).

The ordered list of elements in a geometry may be heterogeneous, i.e., it may be made of elements of different types. A layer is a set of elements sharing the same attributes, thus a layer is represented in one table.

Further, even though Oracle 10g includes a spaghetti as well as a network and a topological models as described Section 3.1, we have chosen the spaghetti model since it is simpler and it represents in a better way intrinsic semantics of multidimensional model. For example, topological models for representing hierarchies guarantees only the so-called "horizontal consistency" [56], i.e., maintaining topological relationships between members belonging to the same level. However, in DWs and OLAP Systems, hierarchies are used for traversing from one level to another and information about the topological relationships between spatial members belonging to the same level is rarely required. Even though some solutions were proposed for representing hierarchies using a topological model (e.g., [56]), i.e., representing in a persistent way the topological relationships between all members belonging to child and parent levels, they include conditions that are too restrictive for our model. For example, imposing that the geometry of every child member must be contained in the geometry of its corresponding parent member and that the geometry of every parent member must be the union of the geometries of the corresponding child members do not allow to model the different kinds of spatial hierarchies.

Additionally, Oracle extends SQL with spatial operators and functions. The difference between both is that the spatial operators use the spatial index while the functions not. For example, the operator sdo.relate allows to test whether two geometries satisfy a topological relationship, while the operator sdo.within_distance tests whether
two geometries are at certain distance from each other. In some situations, functions offer more functionality, for example, sdo.geom.relate unlike sdo.relate allows to determine the name of topological relationship existing between two geometries. Further, Oracle includes the spatial aggregation functions. For example, sdo.aggr.union and sdo.aggr.centroid return, respectively, the spatial union of the given geometries and a geometry which is the centroid.

3.2.3 Preserving semantics

Conceptual models, including the MultiDimER model, provide constructs for representing in a more direct way the semantics of the modeled reality. However, much of this semantics may be lost when translating a conceptual schema into a logical schema since only the concepts supported by the target DBMS can be used. To ensure the semantic equivalence between the conceptual and the logical schemas, integrity constraints can be introduced. The idea of having explicit integrity constraints for spatial data is not new in the spatial database community (e.g., [24, 96, 218]).

Current DBMSs provide support for declarative integrity constraints, such as keys, referential integrity, or check constraints. However, in many cases this support is not sufficient and integrity constraints must be implemented using triggers. A trigger is a named Event-Condition-Action rule that is automatically activated when a table is updated [217]. SQL:2003 as well as major commercial DBMSs support triggers. As a result of using declarative integrity constraints and/or triggers, the semantics of an application domain is kept in the database as opposed to keeping it in the application accessing the database. In this way, constraints are encoded once and are available for all applications accessing the database, enforcing data quality and facilitating application management.

3.3 Spatially-extended MultiDimER model

In this section we briefly present the extension of the MultiDimER model by inclusion of spatial elements. The formal definition of the model can be found in Appendix B. Since the spatially-extended MultiDimER model can contain spatial as well as non-spatial elements, the definitions of a schema, a level, a hierarchy, cardinalities, and a fact relationship remain the same as the ones presented in Section 2.2. Additionally, we define a spatial level as a level for which the application needs to keep its spatial characteristics. This is captured by its geometry, which is represented using spatial data types such as point, line, surface, or a collection of these data types. Usual non-spatial levels are called thematic.

We adopt an orthogonal approach where a level may have geometry independently of the fact that it has spatial attributes. This achieves maximal expressive power where, for example, a level such as State may be spatial or not depending on application requirements, and may have (descriptive) spatial attributes such as Capital (Figure 3.2).

Two consecutive spatial levels forming a hierarchy relate to each other with topological relationships existing between their spatial components, such as contains, equals,
intersects, overlaps, etc. [44]. By default we suppose the contains topological relationship, i.e., the geometry of a parent member contains the geometries of its child members. For representing the geometry of spatial levels as well as topological relationships between them, we use the pictograms of the MADS model [160] as shown in Figure 3.1.

We define a spatial hierarchy as a hierarchy that includes at least one spatial level. Similarly, a spatial dimension is a dimension that includes at least one spatial hierarchy. Usual non-spatial dimensions and hierarchies are called thematic. Spatial hierarchies can combine thematic and spatial levels.

A spatial fact relationship is a fact relationship that requires a spatial join between two or more spatial dimensions. The spatial join can be based on different predicates represented by topological relationships, e.g., intersects. A (spatial) fact relationship may contain measures that can be spatial or thematic. The latter usually represent numerical data meaningful for leaf members that are aggregated while traversing a hierarchy. The former can be represented by geometry or calculated using spatial operators, such as distance, area, etc. To indicate that measure is calculated using spatial operators, we use the symbol (S). Measures require the specification of the function used for aggregations along the hierarchies. By default we suppose sum for the measures represented as numbers and geometric union for the measures represented as geometries.

![Figure 3.2](image_url) An example of a multidimensional schema with spatial elements.

A more detailed description of our notation is given in Section D.4.
3.4 SPATIAL LEVELS

It contains two spatial dimensions, Highway Segment\(^2\) (of type Line) and County (of type Area), as well as two thematic dimensions, Road Coating and Time. Both spatial dimensions contain the levels forming the hierarchies\(^3\). Additionally, the spatial level State includes a spatial descriptive attribute Capital. Since the fact relationship relates all dimensions, it represents a usual join for the thematic dimensions and an intersection topological relationship for the spatial dimensions (the latter represented in Figure 3.2 with the symbol «»). Without considering any measures, the model can answer queries indicating, for example, whether all highway sections pass through some counties or whether some highway sections belong to more than one county. Additionally, the schema contains two thematic measures: No. cars and Repairing cost, and two spatial measures: Length and Common area. Length is a number representing the length of the part of a highway segment that belongs to a county; it is calculated using spatial operators. Common area is a spatial data representing the geometry of this part.

In the following sections, we refer to the different spatial elements of the Multi-DimER model. For each of them, we first present the conceptual model and then, we propose the mapping to the logical level, discussing also implementation considerations.

### 3.4 Spatial levels

#### 3.4.1 Conceptual representation

A spatial level is represented in the MultiDimER model using a geometry symbol next to the level name. For the example in Figure 3.3 a) and b), the surface bag symbol is used for representing the geometry of State members, since States are formed by several counties, some of which may be islands.

![Figure 3.3](image)

Figure 3.3: Examples of a) a spatial level, b) a spatial level with a spatial attribute, and c) a thematic level with a spatial attribute.

A level may have geometry independently of the fact that it has spatial attributes. For example, a level such as State may be spatial (Figure 3.3 a) and b)) or not (Figure 3.3 c)) depending on application requirements, and may have (descriptive) spatial attributes such as Capital (Figure 3.3 b) and c))

\(^2\)Recall that we call a dimension using its leaf level name.

\(^3\)For simplicity we did not include hierarchies in the thematic dimensions.
3.4.2 Logical representation of spatial levels

In the MultiDimER model, a level corresponds to an entity type in the ER model. The spatial support in our model is added in an implicit manner, i.e., the attributes representing the geometry are represented by pictograms. Therefore, the transformation of spatial levels into the classical ER model requires additional attributes for representing their geometries.

Mapping into the object-relational representation will create a table with all attributes specified in the MultiDimER schema and with two additional attributes: one for surrogates and another one for the geometry. The latter represents non-atomic data, which is allowed in the object-relational model. Figure 3.4 a) shows an example of the object-relational representation of the State level from Figure 3.3 a).

Figure 3.4: A spatial level: a) the OR representation of members and b) a member with an island.

The definition of a table representing the State level from Figure 3.3 a) in Oracle 10g Spatial is given next. To ensure the existence of surrogates for the State level, we use an object table, which automatically creates a self-referencing column. The declaration of an object table requires first the definition of a type:

```sql
create type StateType as object (
    Geometry mdsys.sdo_geometry,
    Name varchar2(25),
    Population number(10),
    Area number,
    MajorActivity varchar2(50),
    Capital varchar2(25));
create table State of StateType (
    constraint statePK primary key (Name))
object identifier is System generated;
```

The clause object identifier is system generated indicates that a surrogate attribute is automatically generated by the system (the default option).

The object-relational model of Oracle Spatial provides a unique spatial data type mdsys.sdo.geometry that allows to capture locations and shapes of spatial objects. This is a complex type composed of the following elements:
3.4. SPATIAL LEVELS

- **sdo_gtype** - defines the geometry type, i.e., point, line string, etc.
- **sdo_srid** - is used for identifying a spatial reference system. If null the usual Cartesian system is used.
- **sdo_point** - defines point with a given coordinate X and Y. If the geometry type is different from point, Oracle Spatial ignores this attribute.
- **sdo_elem_info** - is an array of numbers that allows to interpretate the next attribute.
- **sdo_ordinates** - contains an array of the coordinates describing the geometry.

In the example, the attribute Geometry of the mdsys.sdo.geometry data type is used for representing the geometry of a state. The specific geometry (for example, point or line) is defined and instantiated during an insert operation. An example for inserting a state member comprised by two polygons as shown in Figure 3.4 b) is given next:

```sql
insert into State values (
    mdsys.sdo-geometry (2007, null, null,
    mdsys.sdo_elem_info_array(l, 1003,1,11,1003,1),
    mdsys.sdo_ordinate_array (2,5,9,4,8,11,1,9,2,5,10,1,12,2,11,3,9,3,10,1)),
    'WieIkopolska',35000, 45, 'Agriculture', Lublin);
```

The first element of the mdsys.sdo.geometry equal to 2007 defines several components: 2 indicates number of dimensions, 0 refers to a linear referencing system, and 07 represents a multipolygon. The next two elements (two null values) indicate that the Cartesian system is used as a spatial reference system and that the geometry is different from the point type. The following element, sdo_elem_info_array contains two triple values for each of the geometries: 1 and 11 represent the starting positions for coordinates of the next attribute sdo_ordinate_array, 1003 indicates the type of element (1 indicates exterior polygon, the coordinates of which should be specified in counter clockwise order and 3 represents polygon), and the last number of each triple (i.e., 1) specifies that the polygon is represented by straight-line segments. Finally, sdo_ordinate_array contains an array of the coordinates describing the geometry.

When the spatial types defined in the conceptual schema (for example, surface set for the State level in Figure 3.3 a)) are transformed into Oracle Spatial, the semantics may be lost. This may cause that users insert a different spatial data type that the one specified in the conceptual schema. Therefore, to ensure the equivalence for spatial types defined in the conceptual and the logical schemas, in Oracle Spatial a check constraint may be suitable. For example, the following constraint could be included in the schema:

```sql
alter table State add constraint ValidGeom check (Geometry.get_gtype() = 7).
```

However, Oracle does not allow such constructs, thus a trigger must be defined enforcing the geometries of a state to be of the type multipolygon (type 7 in Oracle):

```sql
create or replace trigger ValidGeomState
    before insert or update on State for each row
    begin
        if :new.Geometry.get_gtype() <> 7 then
```
CHAPTER 3. SPATIAL DATA WAREHOUSES

3.4.3 Mapping of levels with spatial descriptive attributes

A spatial level that includes a spatial attribute, as shown in Figure 3.3 b) for the State level, can be mapped to a logical schema in two ways:

1. Include the geometry of a spatial attribute as part of the geometry of a spatial level. For example, the geometry of the State level will include the geometry of the Capital, thus forming an heterogeneous spatial data type, i.e., surface set and point.

2. Represent a spatial level as explained in Section 3.4.2 and additionally include another composite attribute(s) that includes two components: one for representing the value and another one for describing the geometry. For example, the Capital attribute will be composed by capital's name and geometry.

The solution to be chosen depends on users' requirements and implementation considerations. For example, the first solution ensures that the spatial representation of a State always includes a Capital. The second solution allows to include topological constraints explicitly, for example, ensuring that the geometry of a State member contains the geometry of its corresponding Capital member.

The declarations in Oracle 10g Spatial for the State level together with the spatial attribute Capital will slightly change comparing to the previously described ones. For the first option during the insertion the first mdsys.sdo_geometry parameter will be equal to 2004 representing a collection. In the second option, the declaration of the StateType object will include an additional attribute Capital of CapitalType defined as follows:

```sql
create type CapitalType as object (
    Name varchar2(25),
    Geometry mdsys.sdo.geometry);
```

Another case arises when a level is thematic and contains a spatial attribute(s) (Figure 3.3 c)). The mapping to the object-relational model creates a table representing this level containing all its thematic attributes. Further, for every attribute, for which a pictogram indicating its geometry is defined in the MultiDimER schema, an additional composite attribute with two components is included; it comprises an attribute of traditional data types for representing the value (for example, name) and another attribute of the spatial data type for describing the geometry.

3.5 Spatial hierarchies

In the following, we first define spatial hierarchies and then, by means of examples we show that the hierarchy classification proposed in Section 2.4.5 can also be applied for spatial hierarchies. We also present examples of spatial hierarchies forming spatial
3.5. **SPATIAL HIERARCHIES**

Dimensions. Next, we refer to topological relationships existing between spatial levels forming a hierarchy considering their importance for aggregation procedures. Finally, we propose the mapping to the object-relational model taking into account different implementation issues.

3.5.1 **Conceptual representation**

A spatial hierarchy is a hierarchy that includes at least one spatial level. Spatial hierarchies can combine thematic and spatial levels as in Figure 3.12 the Geo location hierarchy. If two consecutive levels forming a hierarchy are spatial, they relate to each other with topological relationships existing between their spatial components. By default we suppose the `contains` topological relationship, i.e., the geometry of a parent member contains the geometries of its child members. For example, the surface bag in Figure 3.5 representing the geometry of the State members contains surfaces representing geometries of its County members.

3.5.1.1 **Different types of spatial hierarchies**

The following definitions of different types of spatial hierarchies are based on the definitions given for thematic hierarchies (Section 2.4). We briefly present them in this part to facilitate the reading and comprehension.

**Simple spatial hierarchies** "Simple spatial hierarchies" are those hierarchies where the relationship between their members can be represented as a tree. Further, these hierarchies use only one criterion for analysis. Simple spatial hierarchies can be further categorized into symmetric, asymmetric, and generalized spatial hierarchies.

![Simple spatial hierarchy diagram](image)

**Symmetric spatial hierarchies** have at the schema level only one path where all levels are mandatory. An example is given in Figure 3.5. At the instance level (Figure 3.5 b)) the members form a tree where all the branches have the same length. As implied...
by the cardinalities, all parent members must have at least one child member and a child member cannot belong to more than one parent member. Notice that different spatial data types are associated to the levels of the hierarchy: point for Store, surface for County, and surface set for State.

Asymmetric spatial hierarchies have only one path at the schema level (Figure 3.6 a)) but, as implied by the cardinalities, some lower levels of the hierarchy are not mandatory. Thus, at the instance level the members represent a non-balanced tree, i.e., the branches of the tree have different lengths (Figure 3.6 b)). The example of Figure 3.6 represents an asymmetric spatial hierarchy for a forest division consisting of little cell, cell, segment, and region. Since some parts of the forest are located in the mountain and are difficult to access, detailed representations of all areas are not available for analysis purposes and some hierarchy members are leaves at the segment or at the cell levels.

Generalized hierarchies contain multiple exclusive paths sharing some levels (Figure 3.7 a)). All these paths represent one hierarchy and account for the same analysis criterion. At the instance level each member of the hierarchy belongs to only one path (Figure 3.7 b)). The symbol ⊗ indicates that for every member the paths are exclusive. In the example, it is supposed that road segments can belong to either city roads or to highways, where the management of city roads is the responsibility of districts while that of highways is privatized. Notice that the geometry associated to the Company level (a surface) represents the spatial extent that a company is responsible for.
3.5. SPATIAL HIERARCHIES

As another example of generalized hierarchies, the data model of the U.S. Census-Administrative Boundaries [202] includes several generalized hierarchies. One of them represents a spatial hierarchy containing a county level. However, in Maryland, Missouri, Nevada, and Virginia the county level is replaced by independent cities or places, whereas in American Samoa, county is replaced by district and islands.

Generalized spatial hierarchies include a special case commonly referred to as non-covering hierarchies. In these hierarchies, some paths skip one or several levels having in common at least the leaf and root levels.

**Non-strict spatial hierarchies** Until now we have assumed that the child-parent links have many-to-one cardinalities, i.e., a child member is related to at most one parent member and a parent member may be related to several child members. However, many-to-many cardinalities are very common in real-life applications related to spatial data, for example, a mobile phone network cell may belong to several ZIP areas [92], several tribal subdivisions in the U.S. Census hierarchy belong both to the American Indian reservation and to the Alaska Native Areas [202].

We call a spatial hierarchy *non-strict* if it has at least one many-to-many cardinality; it is called *strict* if all cardinalities are many-to-one. The members of a non-strict hierarchy form a graph (Figure 3.8 b)). The fact that a hierarchy is strict or not is orthogonal to its type. Thus, the different kinds of spatial hierarchies already presented can be either strict or non-strict.
Figure 3.8: A symmetric non-strict spatial hierarchy: a) schema and b) example of instances.

Figure 3.8 shows a symmetric non-strict spatial hierarchy. The many-to-many cardinality represents the fact that a lake may belong to more than one city. This hierarchy may be used, for example, for controlling the lake contamination level caused by neighboring cities indicating the percentage of the acid, sodium, or dissolved carbon dioxide. Most non-strict hierarchies arise when a partial containment relationship takes place [92], for example, when only part of a lake belongs to a city or only part of a highway belongs to a state. In real situations it is difficult to find non-strict hierarchies with a full containment relationship, i.e., when a spatial member of a lower level wholly belongs to more than one spatial member of a higher level.

Notice that non-strict spatial hierarchies may include a distributing factor symbol $\Theta$ as explained in Section 2.4.2. It indicates how the measure should be distributed between different parent members when a many-to-many cardinality is reached during the roll-up operations.

Multiple alternative spatial hierarchies Multiple alternative spatial hierarchies have several non-exclusive simple spatial hierarchies sharing some levels. However, all these hierarchies account for the same analysis criterion. At the instance level such hierarchies form a graph since a child member can be associated with more than one parent member belonging to different levels. In multiple alternative spatial hierarchies, it is not semantically correct to simultaneously traverse the different composing hierarchies. The user must choose one of the alternative hierarchies for analysis.

The example given in Figure 3.9 represents part of the hierarchies used in the U.S. Census Bureau [202]. The hierarchy for American Indian and Alaska Native Areas, and Hawaii Home Land (AIANA/HHL) uses a particular subdivision of the territory (lower path of the figure). However, the usual hierarchy composed, among others, of County and State levels[^4] (upper path of the figure) provides another subdivision of the territory.

[^4]: To simplify the example, we ignore that some states are not divided in counties.
3.5. SPATIAL HIERARCHIES

Figure 3.9: A multiple alternative spatial hierarchy formed by two non-strict symmetric hierarchies.

The same territory. This path can be used for obtaining statistics of American Indian by counties and states. It is obvious that both hierarchies cannot be simultaneously used during analysis.

Figure 3.10: Parallel independent spatial hierarchies associated to one dimension.

Parallel spatial hierarchies Parallel spatial hierarchies arise when a dimension has associated several spatial hierarchies accounting for different analysis criteria. Such hierarchies can be independent or dependent. In a parallel independent spatial hierarchy, the different hierarchies do not share levels, i.e., they represent non-overlapping sets of hierarchies. An example is given in Figure 3.10.

In contrast, parallel dependent spatial hierarchies, have different hierarchies sharing some levels. The example in Figure 3.11 represents an insurance company that includes hospitalization services for clients. The Client dimension contains two spatial hierarchies. The first one is a symmetric spatial hierarchy representing the hospitalization
structure; it is composed by Client, Hospitalization area, City, Hospitalization region, and State level. The second hierarchy is a non-covering one representing the geographic division of the client’s address; it includes Client, Municipality, City, County (for the States that have it), and State levels. Both spatial hierarchies share the common levels of City and State.

3.5.1.2 Spatial hierarchies forming dimensions

A spatial dimension is a dimension that includes at least one spatial hierarchy. Stefanovic et al. [193] present the definition of spatial dimensions based on spatial references of hierarchy members: non-spatial (a usual thematic dimension), spatial-to-non-spatial (a level has a spatial representation that rolls-up to a non-spatial representation), and fully spatial (all hierarchy levels are spatial). Our definition of spatial dimensions extends that of Stefanovic et al. [193]. Firstly, we allow non-spatial-to-spatial relationships, for example, Client address defined as alphanumeric data type rolls-ups to a spatial City level (Figure 3.12). Secondly, the dimension can be spatial even in the absence of several related levels, i.e., having only one level, for example, a State dimension that is spatial without any other geographical division.

In the example of Figure 3.12 the spatial hierarchy Geo location forms part of the Client dimension, which is also considered as spatial. This example is used for the analysis of clients’ buying behavior. This is realized by establishing selection predicates (using slice-and-dice operations) and by choosing the aggregation level (using roll-up and drill-down operations). These predicates and aggregation levels can be spatial or non-spatial.

The inclusion of spatial dimensions in a multidimensional model not only enriches the query formulation but also can enhance the visualization of the results by displaying

---

5Recall from Section 2.4.1 that the dimension name is derived from its leaf level name.
3.5. SPATIAL HIERARCHIES

Figure 3.12: A spatial hierarchy in the Client dimension.

it in alphanumeric and/or spatial formats. The management and visualization of spatial and non-spatial multidimensional data require a combination of capabilities of OLAP and GIS tools. Several systems, called Spatial OLAP (SOLAP) provide these kinds of manipulations, such as the JMap Spatial OLAP [99].

Additionally, spatial levels forming hierarchies can be shared as in Figure 3.13, where the Client and the Store dimensions share the City and State levels. According to multidimensional normal forms, dimensions should not share levels [110]. However, we consider that this aspect should be managed at the implementation level, at the conceptual level users should be able to reuse levels without unnecessary repetition of their representation.

3.5.2 Topological relationships between spatial levels

Spatial levels forming a hierarchy can be topologically related. The important question to make is which kinds of topological relationships should be allowed between spatial

-In [168] this is called “multiplicity of hierarchy”.
levels considering that these hierarchies are used for aggregating measure values when traversing levels.

For non-spatial hierarchies, summarizability conditions [111] as explained in Section 2.1 must hold for ensuring the correct aggregation of measures in higher levels taking into account existing aggregations in lower levels. Since asymmetric, generalized, and non-strict hierarchies do not satisfy summarizability conditions, it is required to apply either special aggregation procedures (for example, implemented in Microsoft Analysis Services [135] for asymmetric and non-covering hierarchies), or transformations (for example, described in [92] for asymmetric, non-covering, and non-strict hierarchies) as we already discussed in Section 2.4.

Although the summarizability conditions have been established for non-spatial hierarchies they must also hold for spatial hierarchies. However, summarizability problems may also arise depending on the topological relationships existing between spatial lev-
els. Several solutions may be applied: an extreme one is to disallow the topological relationships that cause problems whereas another solution is to define customized procedures for ensuring correct measure aggregation.

We give next a classification of topological relationships according to the required procedures for establishing measure aggregation. Our classification, shown in Figure 3.14, is based on the intersection between the geometric union of the spatial extents of child members (denoted by \(GU(C_{ext})\)) and the spatial extent of their associated parent member (denoted by \(P_{ext}\)).

The \textit{disjoint} topological relationship is not allowed between spatial hierarchy levels since during a roll-up operation the next hierarchy level cannot be reached. Thus, a non-empty intersection between \(GU(C_{ext})\) and \(P_{ext}\) is required.

\[\text{Disjoint, Forbidden} \]
\[\text{Safe aggregation} \]
\[\text{Special aggregation procedure} \]

\[\text{within} \quad \text{equal} \quad \text{Connected} \]
\[\text{Boundary, interior, Both} \]
\[\text{touched, crosses for curves, crosses overlaps} \]

Figure 3.14: Classification of topological relationships for aggregation procedures.

Different topological relationship may exist if the intersection of \(P_{ext}\) and \(GU(C_{ext})\) is not empty. If \(GU(C_{ext}) \text{ within } P_{ext}\), then the geometric union of the child member extents (as well as the extent of each child member) is included in their parent member extent. In this case, the aggregation of measures from a child to a parent level can be done safely using a traditional approach. Similar situation occurs if \(GU(C_{ext}) \text{ equals } P_{ext}\) with the additional constraint that both spatial extents are equal and have common boundaries. Notice that for the \textit{within} topological relationship users must be aware of semantic constraints, for example, having a hierarchy formed by a City and a County spatial levels and a measure Population, the aggregation of all cities' populations does not necessary give the county population, since counties may be formed by other administrative entities. However, for the implementation of aggregation procedures, \textit{within} and \textit{equal} topological relationships can be handled similarly.

The situation is different if the extents of the child and parent members are related by a topological relationship distinct from \textit{within} or \textit{equal}. As can be seen in Figure 3.14 different topological relationships belong to this category, for example, touches, crosses. As in [171], we distinguish three possibilities depending on whether a topological relationship exists between the boundaries, the interiors, or both the boundaries and the interiors of the spatial extents of child and parent members. For example, this

\[\text{We consider the topological relations from the SQL/MM standard [87].}\]
CHAPTER 3. SPATIAL DATA WAREHOUSES

distinction is important in Figure 8 for determining how to realize aggregations if a
lake touches a city and overlaps another.

When developing aggregation procedures, if \( GU(C_{ext}) \) intersects \( P_{ext} \) and this in-
tersection is different from equal or within, the spatial extent of some (or all) child
members is not completely included in the spatial extent of a parent member. There-
due, the topological relationship existing between the spatial extents of every individual
child member and its parent member should be revised. It helps to determine which
measure values can be considered in its entirety for aggregation and which must be
partitioned. For example, if in the hierarchy in Figure 3.5 the geographic union of the
points representing stores is not within the spatial extent of their county, every indi-
vidual store must be analyzed for determining how the measure (for example, required
taxes) should be distributed between two or more counties. Therefore, an appropriate
procedure for measure aggregation according to application particularities must be de-
veloped, such as that proposed by [92] for partial containment topological relationship.
As already said, another solution is to disallow these topological relationships in spatial
hierarchies.

3.5.3 Logical representation of spatial hierarchies

We already proposed in Chapter 2.4 a mapping between non-spatial (i.e., thematic)
levels for different kinds of hierarchies. In this section, we refer to the mapping of
relationships between non-spatial and spatial levels as well as between two spatial
levels. For the latter, we consider the topological relationships existing between the
two spatial levels.

3.5.3.1 Mapping of relationships between levels forming a spatial hierarchy

A relationship between levels forming a hierarchy corresponds to a binary relationship
in the ER model. Therefore, this relationship can be represented in the object-relational
model using the traditional mapping for the binary many-to-one relationships as de-
scribed in Section 2.3. This requires to include in the table created for the child level
an attribute for representing a parent key. Notice that this mapping does not depend
on whether the levels are spatial or not, i.e., it will be the same when one level or
both are spatial. For example, the mapping of the relationship between the County
and State levels in Figure 3.15 and 3.16 gives the County table with an additional
attribute, which contains the surrogates from the State table.

![Diagram of County and State relationship]

Figure 3.15: Example of the relationship between non-spatial and spatial levels.
Using Oracle 10g Spatial a table for the State level will be created in a similar way as explained in Section 3.4.2. The definition of a table for the County level in Figure 3.15 is as follows:

```sql
create type CountyType as object (
    Name varchar2(25),
    Population number(10),
    Area number,
    StateRef ref StateType);
create table County of CountyType (
    StateRef NOT NULL,
    constraint CountyPK primary key (Name),
    constraint CountyFK foreign key (StateRef) references State);
```

The CountyType object includes a reference (ref) type that points to the corresponding row in the State table. In this way, the object-relational approach replaces value-based joins with direct access to related rows using the identifiers. Further, not allowing the attribute StateRef to have null values and enforcing referential integrity constrains ensure that every county member will have assigned a valid state member. However, when data is inserted into the County table, the surrogates of the corresponding state members should be known. To facilitate this operation we first create a view allowing the user to introduce a state name instead of a state surrogate:

```sql
create view CountyView(Name,Population,Area,StateName) as
    select C.Name,C.Population,C.Area,S.Name
    from County C, State S
    where C.StateRef = ref(S);
```

Since views defined on two tables cannot be updated, to insert data into the County table using the CountyView, for example using:

```sql
insert into CountyView values ('Countyl', 100000, 3000, 'StateA');
```

an instead of trigger should be created. This kind of triggers can only be used for views and it performs actions instead of the operation specified in the trigger. The following trigger first checks if the state name exists in the State table and then performs the corresponding actions for inserting the reference in the County table or raises an error message otherwise:

```sql
create or replace trigger CountyIns instead of insert on CountyView
for each row
declare
    NumRows number(5);
begin
    select count(*) into NumRows
    from State S
    where :new.StateName = S.Name;
    if NumRow = 1 then
        insert into County
        from State S
    end if;
end;
```

where S.name = :new.StateName;
else
    raise_application_error(-2000, 'lnvalid State Name: ' || :new.StateName);
end if;
end;
Similar triggers can be created for the update and delete operations to facilitate operations and to ensure data integrity.

3.5.3.2 Representing topological relationships between two spatial levels

![Diagram of the relationship between two spatial levels: State and County.](image)

**Figure 3.16:** Example of relationship between two spatial levels.

Topological relationships between spatial levels forming a hierarchy should also be considered at the logical level for preventing the inclusion of incorrect data and for indicating what kind of aggregation procedures should be developed. Two solutions can be proposed: (1) constrain the geometry of the child member during the insert operation or (2) verify topological relationships between the geometric union of the spatial extents of child members and the spatial extent of their associated parent member, after the insertion of all child members.

We explain next both solutions. For this we require to create tables that represent child and parent spatial levels and the relationships between them (for example, for the County and State levels in Figure 3.16) as explained in Section 3.5.3.1. Then, we create a view (CountySpaView) similar to the previously defined view (CountyView) for facilitating data insertions:

```sql
create view CountySpaView(Geometry,Name,Population,Area,StateName) as
    select C.Geometry, C.Name,C.Population,C.Area,S.Name
    from County C, State S
    where C.StateRef = ref(S);
```

The first solution, which constrains the geometry of the child member during the insert operation, requires the extension of the previously-defined instead of trigger (CountyIns) for verifying the topological relationship between spatial extents of a county and a state members:

```sql
create or replace trigger CountySpaIns instead of insert on CountySpaView
for each row
declare
    StGeometry State.Geometry%Type;
begin
    select S.Geometry into StGeometry
```

```sql
end;
```

3.5. **SPATIAL HIERARCHIES**

from State S
where S.Name = :new.StateName;
if SQL%found then
    if sdo.geom.relate(S.Geometry,'anyinteract',new.Geometry,0.005) = 'TRUE' then
        insert into County
        from State S
        where S.name = :new.StateName;
    else
        raise_application_error(-2002, 'Invalid Disjoint Topological Relationship');
    end if;
else
    raise raise_application_error(-2000, 'Invalid State Name: ' || :new.StateName);
end if;
end;

The trigger raises errors if the state name is invalid or if the geometry of a county member is disjoint from the geometry of its corresponding state member. Otherwise, it inserts the new data into the County table. In the example, to check topological relationships we use the `sdo.geom.relate` function with an 'anyinteract' mask, which accepts any topological relationships but disjoint between child and parent members. However, specific topological relationship can be used instead of anyinteract, for example, covers\(^8\) as in:

\[
\text{sgo.geom.relate(S.Geometry,'covers',new.Geometry,0.005) = 'covers'};
\]

The value 0.005 in the `sdo.geom.relate` operator indicates tolerance, which reflects the distance that two points can be apart and still be considered the same.

In the second solution we allow to include child members without activating an instead of trigger. After all child members are inserted, the verification of the topological relationship between the geometric union of the spatial extents of child members and the spatial extent of their associated parent member is performed. An example of this verification is given next. First, we define a function that receives a State name and returns 1 if the spatial extent of a given State member is equal to the geometric union of the spatial extents of its County members:

\[
\text{create or replace function ChildrenWithinParent (StateName State.Name%Type) return Number is}
\]

\[
\text{StName State.Name%type; begin select SI.Name into StName from State SI, ( select S2.Name as SName, sdo.aggr.union(sdoaggrtype(C.Geometry, 0.005)) as Geometry from County C , State S2 where C.StateRef = ref(S2) group by S2.Name ) GU}
\]

\[^{8}\text{It returns covers, if the second object is entirely within the first object and the boundaries touch in one or more places.}\]
where S1.Name = StateName and GU.SName = S1.Name and
sdo_geom.relate(S1.Geometry, 'equal', GU.Geometry, 0.005) = 'equal';
if SQL%found then return 1;
else return 0;
end if;
end;

We use the sdo_aggr_union function, which returns a spatial object represented as
the géométrie union of the specified spatial objects, for example, county members.
This function works similarly to the aggregate functions used for non-spatial data, i.e.,
when the group by clause is included and the specific function is selected (for example,
sum) with the difference that it refers to spatial data. The select statement in the from
clause, creates a temporary table GU with two attributes SName and Geometry. The
latter is the géométrie union of counties grouped by a state name. Then, this table is
used in the second where statement for testing the equal topological relationship.

The ChildrenWithinParent function can be called for a specific state or for all states.
Next, we show an example of this call displaying a message instead of taking some
specific action:

```sql
declare
    StName State.Name%type;
cursor RetrieveState is
    select S.Name
    from State S;
begin
    open RetrieveState;
    loop
        fetch RetrieveState into StName;
        exit when RetrieveState%notfound;
        if (ChildrenWithinParent (StName) = 1) then
            dbms_output.put_line(StName || ' is totally covered by its counties');
        else
            dbms_output.put_line(StName || ' is not totally covered by its counties');
        end if;
    end loop;
    close RetrieveState;
end;
```

Since the branch else indicates that some (or all) counties intersect their state
member⁶, we must check the topological relationships of individual child members.
The topological relationship can be easily retrieved in Oracle using, for example, the
following function for a state member S in the State table and every related child
member C in the County table:

```
sdo_geom.relate(S.Geometry, 'determine', C.Geometry, 0.005);
```

In the following example, we expand the else statement of the previous example.

⁶In real situation, counties are included in states, i.e., this topological relationship is equal, however we use the same example to shorten the chapter size.
3.6. SPATIAL FACT RELATIONSHIPS AND MEASURES

For every state member we check the existing topological relationships with its county members, displaying the names of the state, of the county, and of the topological relationship existing between them, for example, StateA contains County2:

```sql
declare
    StName State.Name%type;
    TopRelName varchar2(30);
    CountyName CountySpa.Name%type;
    cursor RetrieveState is
        select S.Name
        from State S;
    cursor RetrieveCounty (StName2 State.Name%type) is
        select C.Name,
            sdo_geom.relate(S.Geometry, 'détermine', C.Geometry,0.005) relationship
        from CountySpa C, State S
        where C.StateRef = ref(S) and S.Name=StName2;
begin
    open RetrieveState;
    loop
        fetch RetrieveState into StName;
        exit when RetrieveState%NOTFOUND;
        if (Children_Within_Parent (StName) = 1) then
            dbms_output.put_line(StName || ' is totally covered by its counties');
        else
            dbms_output.put_line(StName || ' is not totally covered by its counties');
        end if;
        open RetrieveCounty(StName);
        loop
            fetch RetrieveCounty into CountyName .TopRelName;
            exit when RetrieveCounty%NOTFOUND;
            dbms_output.put_line(StName || TopRelName || CountyName);
        end loop;
        close RetrieveCounty;
    end loop;
    close RetrieveState;
end;
```

3.6 Spatial fact relationships and measures

In the next sections we first refer in more detail to spatial fact relationships and spatial measures. We present several examples of spatial multidimensional schemas that show usefulness of having applicability of these spatial elements in the DW context. We also include mapping to an object-relational model and refer to implementation issues.
CHAPTER 3. SPATIAL DATA WAREHOUSES

3.6.1 Conceptual representation

In a multidimensional model a fact relationship links leaf members from all dimensions participating in the relationship. For non-spatial dimensions this relationship corresponds to the relational join operator. However, if there are more than one spatial dimension, a topological relationship linking the different spatial levels is required.

A (spatial) fact relationship may contain measures that can be spatial or thematic. The latter usually represent numeric data. The former can be represented by geometry or calculated using spatial operators. Next, we refer in more detail to spatial fact relationships and spatial measures.

3.6.1.1 Spatial fact relationships

![Figure 3.17: Example of schema for analysis of the city transportation service.](image)

Examples of spatial fact relationships with the intersect topological relationship are shown in Figure 3.2 and Figure 3.17. The example in Figure 3.2 was already described in Section 3.3. In this example, two spatial dimensions participate in a fact relationship, thus, the latter includes an intersection topological relationship representing the predicator used for the spatial join operation. It indicates that users require to focus their analysis on those highway segments that intersect counties.

Another example of a spatial fact relationship is shown in Figure 3.17. This schema is used to improve the transportation service system in a city and to decrease the number of overlapping routes of different kinds of transportation. The city is divided in sections and for each of them the bus, metro, and tramway maps are developed. The spatial fact relationship representing the intersections of these transportation lines includes additional numeric measures of the number of stops and a spatial measure Common area, to which we refer in the next section.

Notice that spatial DWs may include a feature not commonly found in spatial DBs: n-ary topological relationship, as shown in Figure 3.17. In spatial DBs, the topological relationships are usually represented as binary relationships. However, in spatial DWs where the focus is topological analysis, the topological relationships of interest can include more than two spatial dimensions. This n-ary topological relationship is not a trivial task and has not been extensively researched. One treatment of this kind of relationship is given in [171].

[^10]: For simplicity the hierarchies are not taken into account.
3.6. SPATIAL FACT RELATIONSHIPS AND MEASURES

3.6.1.2 Spatial measures represented by a geometry

Spatial measures represented by geometries are included in Figure 3.2 and Figure 3.17. In the former, the Common area measure represents the geometry (line) of part of a highway segment that belongs to a county. In the latter, it represents the geometry resulting from intersections of different transportation lines. Further, the resulting spatial measure in Figure 3.17 is represented by alternative geometries (point or line) as indicated by the symbol •.

Including spatial measures requires defining their management during aggregations when the roll-up and drill-down operations are applied to hierarchies. Current OLAP systems automatically aggregate numerical measures along hierarchies using different types of functions: distributive, algebraic, and holistic [63]. For example, distributive functions, such as sum, min, and count, reuse aggregates of a lower level of a hierarchy in order to calculate the aggregates for a higher level. Algebraic functions, such as average, variance, and standard deviation, need an additional treatment for reusing the values, for example, the average of a higher level can be calculated taking into account already computed values of sum and count of a lower hierarchy level. On the other hand, holistic functions, such as median, most frequent, and rank require new calculations using the data of the leaf level.

As for non-spatial data, aggregation functions for spatial data have been defined [186]. For example, spatial distributive aggregates include convex hull, geometric union, and geometric intersection. The result of these functions is represented by simple or complex geometries. Examples of spatial algebraic functions are center of n geometric points or center of gravity, while examples of spatial holistic functions are equi-partition...
or nearest-neighbor index [186].

In current OLAP systems an aggregation function must be specified for numeric measures. If a function is not specified, the sum will be applied by default. Similarly, if a spatial multidimensional model contains measures represented by a geometry, it should also include the spatial aggregation function that will be used when the roll-up and drill-down operations are applied to some hierarchies (spatial or thematic). Therefore, when the hierarchies are included in the schema, our model requires the inclusion of the spatial aggregation function for spatial measures represented by geometry. This specification may not be included if the geometric union is used.

Table 3.1: Examples of queries for conventional DWs and spatial DWs.

<table>
<thead>
<tr>
<th>Queries</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sales Model</strong></td>
<td><strong>Accident Model</strong></td>
</tr>
<tr>
<td>Total sales in store X of products of category Y in year Z.</td>
<td>Locations where a client X had accidents covered by an insurance of category Y in year Z.</td>
</tr>
<tr>
<td>In both models, the predicates for the corresponding dimensions (store X, client X) are applied; the measure is aggregated taking into account product category or insurance category and year; a sum of sales or a spatial union is performed, respectively.</td>
<td></td>
</tr>
<tr>
<td>Total sales of products of category X.</td>
<td>Locations of accidents covered by insurances of category X.</td>
</tr>
<tr>
<td>Similar to the previous case; the Client, Store, and Time dimensions are considered at the highest level (for all clients, all stores, and all times).</td>
<td></td>
</tr>
<tr>
<td>Total sales in year X grouped by city.</td>
<td>Locations of accidents in year X grouped by client age group.</td>
</tr>
<tr>
<td>The grouping by city or by age group is performed; then, the combination of measure values follows: a sum for sales and a spatial union for the location of accidents.</td>
<td></td>
</tr>
<tr>
<td>Total sales for the product categories that were sold more than 1000 times in year X.</td>
<td>Locations where more than 20 car accidents occurred in year X.</td>
</tr>
<tr>
<td>In both cases, the fact relationship is scanned and objects are grouped according to product categories or insurance categories, respectively; then, the occurrence of the item (product or location) is calculated in order to check the condition.</td>
<td></td>
</tr>
</tbody>
</table>

In the multidimensional schema in Figure 3.18, the user is interested in analyzing locations of accidents taking into account the different insurance categories (full coverage, partial coverage, etc.) and particular client data. The model includes a spatial measure representing the location of an accident. Since the dimensions include hierarchies, a spatial aggregation function is defined, in this case the geometric union (GU).

\[11^{11}\text{To make the query more realistic we can use a buffering operation for location: there are not many accidents that occur exactly at the same place.}\]
Thus, when a user rolls-up to the Insurance category level, the locations corresponding to different categories will be aggregated and represented as a set of points applying different display techniques, such as different colors. Other spatial operators can be also used, for instance, center of n points. Notice that the example includes feature that is not considered by existing proposals in spatial DWs: a spatial measure without spatial dimensions. The usual practice in spatial DWs is to include spatial measure only if spatial dimensions are present. Nevertheless, we adopt a different approach allowing measures to be spatial independently of the fact that spatial dimensions are included in the model.

![Spatial Dimension: another variant of a schema for analysis of accidents.](image)

In order to show the importance of using a multidimensional model for spatial data representation we analyze different queries that can be posed for models with non-spatial and spatial measures. Table 3.1 compares some OLAP queries for the sales model with non-spatial measures of Figure 2.3 with equivalent queries for the model of analysis of accidents of Figure 3.18, which contains a spatial measure. We suppose that the Client and Insurance dimensions in Figure 3.18 correspond to the Store and Product dimensions in Figure 2.3, respectively. We also, include some comments about the similarity of the actions performed for answering the queries.

Table 3.1 shows similar queries with different kinds of measures: spatial and non-spatial. This list of queries can be extended by the inclusion of the other non-spatial
measures, such as amount of paid insurance. As shown by the examples, the analysis using spatial measures is richer than using non-spatial representation; for example, a union of different locations that represents accidents is more informative than aggregating non-spatial measures from the fact relationship without taking into account any other attributes. Retrieving a map representation for the obtained locations can show regions of a city with high accident factor; in contrast, retrieving product numbers or names still needs additional processing for analysis. Here, we assume that no additional treatment of measures is done (characterization, clustering, association, etc.) as the ones presented in [78].

<table>
<thead>
<tr>
<th>Queries</th>
<th>Models for the analysis of accidents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spatial Measure</strong> (Figure 3.18)</td>
<td><strong>Spatial Dimension</strong> (Figure 3.19)</td>
</tr>
<tr>
<td>Visualization of locations of accidents.</td>
<td>Can be done displaying the location measure in a map.</td>
</tr>
<tr>
<td>Aggregate locations (geometric union) of accidents according to some predicate involving time or insurance.</td>
<td>Can be done when a roll-up operation in the time or insurance hierarchy is executed.</td>
</tr>
<tr>
<td>Comparison of amount of insurance paid in different geographic zones.</td>
<td>Cannot be done. Only the exact location of accidents is known in the system without their geographic distribution by city, state, etc. If the buffering of location can be used, some locations can be considered together. However, they may not correspond to established geographic zones.</td>
</tr>
<tr>
<td>Amount of particular insurance paid in some specific geographic area.</td>
<td>Cannot be done. Only specific locations are given. The aggregation of location is done according to other hierarchies.</td>
</tr>
</tbody>
</table>

An alternative model for the analysis of accidents is shown in Figure 3.19; it does not include a spatial measure and the focus of analysis has been changed to the amount

\[12\] We do not discuss here the manipulation of the query result, for example, windows query.
3.6. SPATIAL FACT RELATIONSHIPS AND MEASURES

of insurance paid according to different geographical locations. Furthermore, since it contains a spatial hierarchy, it allows spatial data manipulation in the predicator or grouping roles using slice-and-dice, drill-down, and roll-up operations.

Table 3.2 presents some characteristics that differentiate both models with respect to the queries that can be addressed to each one. Although these models are similar, different analysis can be made when a location is handled as a spatial measure or as a spatial hierarchy. It is the designer's decision to determine which of these models better represents users' needs.

Another feature that distinguishes the two models given in Figures 3.18 and 3.19 is pre-aggregation. Pre-aggregation means to store persistently the aggregated measure values according to different hierarchy levels in order to accelerate the execution of the roll-up operations, i.e., creating so-called materialized views. Therefore, spatial or non-spatial measures introduced in fact relationships can be pre-aggregated according to the existing hierarchies. The decision of which aggregates are pre-calculated is based on several criteria and there are different algorithms developed for non-spatial (e.g., [177]) and spatial (e.g., [155, 193]) measures.

On the other hand, the hierarchy levels (whether spatial or non-spatial) represent different degrees of generalization and all these levels are represented in a system a priori without the necessity to incur into additional calculations.

3.6.1.3 Spatial measures as result of spatial computation

Spatial measures do not need to be represented by a geometry, they may be obtained by applying spatial or topological operators. The example in Figure 3.2 includes the Length measures, which is a number representing the length of a highway segment that belongs to a county. In the figure symbol (S) allows to distinguish this kind of measures. Another example in Figure 3.20 includes a measure that indicates minimum distance from cities to highway segments. In both models (Figure 3.2 and 3.20), the measure is represented as numbers, nevertheless they require a spatial operator for their calculation.

Further, similar to non-spatial DWs, if spatial calculated measures are aggregated according to hierarchy levels, they must be classified, i.e., it must be specified if they are distributive, algebraic, or holistic before the aggregation takes place [186]. For example, the measure Length in Figure 3.2 is distributive, i.e., it can be aggregated taking into account the length calculated for a lower hierarchy level. However, if the measure represents the percentage of the area that a highway segment occupies in relation to the county area, this measure cannot be aggregated for the State level reusing existing measures for the County level. Another measure Min. distance in Figure 3.20 is holistic: the minimum distance from states to highway sections cannot be obtained taking into account minimum distances between cities and highway segments.

In order to exploit the similarity between OLAP and SOLAP systems, we ignore the issue of dynamically-created aggregations through windows queries [156].

For simplicity, we either ignore that states can have highway sections that do not belong to cities or include an additional member "other" that contains the total length of sections, which belong to state but do not pass through cities.
CHAPTER 3. SPATIAL DATA WAREHOUSES

3.6.2 Mapping of spatial fact relationships

A fact relationship in the MultiDimER model corresponds to an n-ary relationship in the ER model. Thus, using the traditional mapping to relational databases we represent the fact relationship as a separate table with foreign keys of leaf members participating in this relationship and additionally include all attributes represented in the model as measures.

An example of spatial fact relationship was given in Figure 3.2. In this section we refer to this example not considering for now the spatial measures Length and Common area. Suppose that levels and hierarchies are already defined according to the previous explanations. In Oracle 10g, we can define a table for the fact relationship as follows:

```sql
create table HighwayMaintenance (    HighwaySegmentRef ref HighwaySegmentType,    RoadCoatingRef ref RoadCoatingType,    CountyRef ref CountyType,    TimeRef ref TimeType,    NoCars integer,    RepairCost real,    /* foreign key constraints */);
```

We do not use an object table in the declaration above since in our model the fact relationship does not exist without their corresponding levels.

The spatial join in the logical schema states that the members of spatial dimensions can be included in the fact relationship table only if their geometries satisfy the topological relationship specified in the conceptual schema, for example, intersection in Figure 3.2. The following statement can form part of a trigger or an application for loading data allowing to select only those surrogates of HighwaySegment and County tables for which geometries are not disjoint:

```sql
select ref(HS), ref(C)    from HighwaySegment HS, County C
```
3.6. Spatial Fact Relationships and Measures

In Oracle Spatial, sdojoin is not an operator but a so-called table function. This function is recommended when full table joins are required, i.e., each of the geometries in one table is compared with each of the geometries in the other table. The function sdojoin returns a table that contains pairs of row identifiers (rowid1, rowid2) from participating tables that satisfy the topological relationship specified in the mask. In the example, we specify anyinteract, however other topological relationship can also be used, for example, covers.

After this validation, additionally different spatial analysis can be provided based on the existing topological relationship between spatial dimensions. The following example retrieves the geometry definition of a common area and the type of topological relationship existing between spatial members of two leaf levels, respectively:

```sql
select HS.Number, C.Name,
       sdo_geom.sdo_intersection(HS.Geometry, C.Geometry, 0.005),
       sdo_geom.relate(HS.Geometry, 'determine', C.Geometry, 0.005) relationship
from HighwaySegment HS, County C, HighwayMaintenance HM
where HM.HighwaySegmentRef = ref(HS) and HM.CountyRef = ref(C);
```

However, if more than two spatial dimensions participate in the spatial fact relationship as in Figure 3.17, it may be more difficult to implement the restriction to include in the fact table only surrogates of those leaf levels which spatial component satisfy the topological relationship specified in the conceptual level. For example, Oracle Spatial does not allow spatial join between more than two attributes of spatial data type. Therefore, the application should be responsible of handling this operation.

On the other hand, different operators can be useful for retrieving the identifiers of three spatial objects having some common area as in the following example:

```sql
select ref(S), MT.MetroLineRef, MT.TramwayLineRef
from Street S,
     (Select ref(ML) as MetroLineRef, ref(TL) as TramwayLineRef,
      sdo_geom.sdo_intersection(ML.Geometry, TL.Geometry, 0.005) as Geometry
      from MetroLine ML, TramwayLine TL) MT
where sdo_anyinteract(S.Geometry, MT.Geometry) = 'TRUE';
```

3.6.3 Mapping of Spatial Measures

3.6.3.1 Spatial Measures Represented by a Geometry

The mapping of spatial measures, such as the Common area measure in Figure 3.2 or the Location measure in Figure 3.18 requires to include an additional attribute of spatial data type of line for the former and of point for the latter.

The specific geometries for spatial measures can be inserted by giving the corresponding coordinates, as for the Location measure, or by deriving them from the geometries of participating level members. For the latter, the geometries representing the intersection of geometries of Metro line, Tramway lines, and Street members for the example in Figure 3.17 can be obtained as follows:
select sdo.geom.sdo_intersection(S.Geometry, MT.Geometry, 0.005) from Street S, 
(Select ref(ML) as MetroLineRef, ref(TL) as TramwayLineRef, 
sdo.geom.sdo_intersection(ML.Geometry, TL.Geometry, 0.005) as Geometry 
from MetroLine ML, TramwayLine TL) MT 
where sdo.anyinteract(S.Geometry, MT.Geometry) = 'TRUE';

An interesting aspect for this kind of measures is the aggregation during the roll-up operation. A simplifying example of the aggregation is given next for the schema in Figure 3.18. For this example we suppose that the fact relationship table already exists and we also use the notations similar to the previous examples, for example, A.ClientRef is the reference to rows of a Client table from a corresponding table A.

select AG.GroupName, sum(A.AmountPaid), sdo_aggr_union(sdoaggrtype(A.Location, 0.005)) from Accidents A, Client C, AgeGroup AG 
where A.ClientRef = ref(C) and C.AgeGroupRef = ref(AG) 
group by AG.GroupName;

The roll-up operation as shown above is applied only to the Client dimension without any restrictions for other dimensions. However, if roll-up is applied for more than one dimension, for example, Client and Time, the group by rollup operator can be used:

select AG.GroupName, M.Name, sum(A.AmountPaid), 
sdo_aggr_union(sdoaggrtype(A.Location, 0.005)) from Accidents A, Client C, AgeGroup AG, Time T, Month M, 
where A.ClientRef = ref(C) and A.TimeRef = ref(T) and 
C.AgeGroupRef = ref(AG) and T.MonthRef = ref(M) 
group by rollup (AG.GroupName, M.Name);

This query gives the sum of amount paid and the geometric union of the locations where accidents have occurred according to different age group names and months. For example, if we have 3 age groups called G1, G2, G3, and month May and June, aggregated measures are specified for combinations (G1, May), (G2, May), (G3, May), (G1, June), (G2, June), and (G3, June). Additionally, the subtotal for each age group is given, for example, for G1, G2, and G3 as well as the grand total.

Another operator group by cube returns subtotals for the so-called cross-tabulation considering all combinations of grouping members presented in the selected dimensions. For the previous example, the same totals could be calculated including additionally totals for May and June. However, in the current version of Oracle Spatial this operator gives an error.

Notice that different spatial aggregation functions can be used. For example, a user can require center of n points instead of geometric union that can be present in Oracle Spatial using:

sdo_aggr_centroid(sdoaggrtype(A.Location, 0.005)).

3.6.3.2 Calculated spatial measures

An example of a spatial calculated measure was given in Figure 3.2, i.e., the Length measure. Notice, that this spatial measure may be calculated considering the spatial dimensions as shown in the figure or spatial dimensions may be not included in the
model and the measure is calculated based on spatial data in source systems. These kinds of measures are considered spatial since they are obtained by applying spatial or topological operators, even though they are represented using traditional data types, for example, real or integer. In Oracle 10g this measure can be obtained as follows:

\[ \text{sdgeom.sdo.length(sdo.geom.sdo.intersection(H.Geometry, C.Geometry, 0.005), 0.005)} \]

where H and C indicate the HighwaySegment and the County tables, respectively.

This measure can be aggregated according to a type of the aggregated function as specified in Section 3.6.1.2.

### 3.7 Metamodel of the spatially-extended MultiDimER model

In this section we present the metamodel of the spatially-extended MultiDimER
model as shown in Figure 3.21. A dimension comprises either a level, or one or more hierarchies. A hierarchy can belong to only one dimension. It includes an analysis criterion, which expresses different structures used for analysis, for example, geographical location, organizational structure. A hierarchy contains several related levels and these levels may be shared between different hierarchies. Levels may be spatial.

A spatial level is characterized by its geometry (the Geometry attribute in Figure 3.21), which is represented using spatial data types or a collection of these data types. Levels include attributes, some of which are key attributes used for aggregation purposes. Other attributes are descriptive and may be of spatial data types.

Levels forming a hierarchy are associated through the child-parent relationship (the Connects relationship in Figure 3.21). This relationship is characterized by the cardinalities. Cardinalities indicate the minimum and maximum numbers of members in a child level that can be related to a member in a parent level. Further, if two related levels are spatial they can be related topologically (TopRel attribute in the Connects relationship in Figure 3.21).

A dimension is spatial if it has at least one spatial hierarchy or level. A hierarchy is spatial if it has at least one spatial level. Since spatial hierarchies (respectively dimensions) can combine spatial and thematic levels (respectively hierarchies), we call a hierarchy (respectively dimension) fully spatial when all its levels (respectively hierarchies) are spatial. It is called partly spatial when it contains at least one thematic level (respectively one thematic hierarchy).

A fact relationship represents an n-ary relationship between leaf levels with $n > 1$. If the two or more levels are spatial, this relationship may also be topological and requires the inclusion of the spatial join predicate. The fact relationship may contain attributes commonly called measures. In the MultiDimER model measures can be spatial or thematic. The latter are numeric and express analysis needs in a quantified form. The former can be represented by a geometry or by a number, which is calculated using spatial operators.

Notice several differences between this metamodel and the metamodel presented for conventional DWs (Figure 2.4):

- The inclusion of the derived attribute called Spatial for Dimension (respectively for Hierarchy) allows to store the information whether a dimension (respectively a hierarchy) is spatial.

- Levels as well as attributes have an additional attribute (Geometry) representing their geometries if they are spatial. Notice since measures are attributes of fact relationships, the additional specialization is included in the metamodel allowing measures to be represented by geometry or calculated.

- The presence of topological relationships between child and parent levels forming a hierarchy is represented in an additional attribute called TopRel in the relationship Connects.

- A fact relationship is specialized in two types: spatial and thematic, where the former requires the inclusion of a topological operator indicating the topological relationship between spatial leaf levels participating in this fact relationship.
3.8 Related work

Spatial databases (SDBs) have been investigated over the last decades (e.g., [74, 45]). Different aspects are considered, such as conceptual and logical modeling, specification of topological constraints, query languages, spatial index structures, efficient storage management, etc. Rigaux et al. [178] refer in more details to these and other aspects of spatial database research. Further, Viqueira et al. [208] present an extensive evaluation of spatial data models considering spatial data types, data structures used for their implementation, and spatial operations for GIS-centric as well as DBMS-centric architectures.

Further, even though there are works that focus on conceptual modeling for spatial DBs (e.g., [159]) and for DWs (e.g., [199]) based on either the ER model or the UML, a multidimensional model is seldom used for spatial data modeling. Moreover, although organizations such as ESRI recognize the necessity of conceptual modeling by introducing templates of spatial data models in different areas of human activities [52], these models often refer to particular aspects of the logical-level design and are too complex to be understood by decision-making users.

To our knowledge, very few conceptual models based on a multidimensional view of spatial data were proposed, e.g., [92, 162]. Pedersen and Tryfona [162] extend the work of Pedersen et al. [161] by inclusion of spatial measures. They focus on the problems of aggregations in the presence of different topological relationships existing between spatial measures.

On the other hand, Jensen et al. [92] extend the model proposed by Pedersen and Tryfona [162] allowing to include spatial objects in hierarchies with partial containment relationships, i.e., where only part of spatial object belongs to a higher hierarchy level (for example, only part of a street belongs to a higher hierarchy level represented by a ZIP area). They also classify spatial hierarchies based on the work of Pedersen et al. [161]. However, in their proposal it is not clear whether or not levels are spatial, i.e., the difference between partly and fully spatial hierarchies cannot be made. They mostly focus on inaccuracy existing in aggregation paths and to the transformations of hierarchies with partial containment relationships to simple hierarchies.

Other works consider the integration between GIS and DW/OLAP environments. Pourabbas [166] and Ferri et al. [54] refer to common key elements between spatial and multidimensional databases: time and space. They formally define a geographic data model including contains and the full-contains relationships between hierarchy levels. Based on these relationships, the integration between GIS and DW/OLAP environments can be achieved by a mapping between the hierarchical structures of both environments. Moreover, based on full-contains function they are able to provide data lacking in one environment using data from another environment.

The concept of mapping between hierarchies is also exploited by Kouba et al. [105]. To ensure the consistent navigation in a hierarchy between OLAP systems and GISs, they propose a dynamic correspondence through classes, instances, and action levels defined in a meta-data repository.

Several authors define elements of spatial DWs, i.e., spatial measures and dimensions. For example, Stefanovic et al. [193] propose three types of spatial dimensions based on the spatial references of the hierarchy members: non-spatial (a usual thematic...
CHAPTER 3. SPATIAL DATA WAREHOUSES

hierarchy), spatial-to-non-spatial (a level has a spatial representation that rolls-up to a non-spatial representation), and fully spatial (all hierarchy levels are spatial). We consider that non-spatial-to-spatial references should also be allowed since a non-spatial level (for example, address represented as alphanumeric data type) can roll-up to a spatial level. Further, we extended the classification for spatial dimensions allowing the dimension to be spatial even in the absence of several related levels, i.e., having only one spatial level.

Regarding measures, Stefanovic et al. [193] distinguish numerical and spatial measures; the latter represent the collection of pointers to spatial objects. Rivest et al. [179] extend the definition of spatial measures and include measures represented as spatial objects or calculated using spatial metric or topological operators. However, in their approach the inclusion of spatial measures represented by a geometry requires the presence of spatial dimensions. On the contrary, in our model a spatial measure can be included in the model with only thematic dimensions (Figure 3.18).

On the other hand, Fidalgo et al. [55] exclude spatial measures from SDWs. Instead, they create spatial dimensions that contain spatial objects previously-represented as measures. They extend a star schema including two new dimensions for managing spatial hierarchies: geographical and hybrid. Both dimensions are subsequently divided in more specialized structures. However, their proposal have several drawbacks. For example, they do not allow to share spatial objects represented by a point. This is very restrictive for some kinds of applications, for example, a city represented by a point that is shared between Store and Client dimensions. Further, we consider that spatial measures should be allowed in SDWs and we already showed several spatial DW scenarios that justify their inclusion (Section 3.6).

The proposed extensions by Stefanovic et al. [193], Rivest et al. [179], and Fidalgo et al. [55] are mainly based on the star schema. This logical representation lacks expressiveness not allowing to distinguish spatial and non-spatial data and to include topological constraints as the ones proposed in our model. Further, none of the current works considers spatial fact relationships as proposed in this thesis.

The work of van Oosterom et al. [203] evaluates different DBMSs that include spatial extensions, i.e., Oracle, Informix, and Ingres, with respect to their functionality and performance. The functionality is compared to the Simple Feature Specification (SFS) for SQL defined by the OpenGIS Consortium (OGC) [145] and the performance is evaluated by inserting and querying high volumes of spatial data. They conclude that currently spatial DBMSs are sufficient for storage, retrieval, and simple analysis of spatial data. Further, they state that Ingress has the least richness with respect to the functionality, that Informix is the only one compliant to the OpenGIS Specifications, and that Oracle does not always have the best performance, but it has the richest functionality.

Concepts used for multidimensional modeling applied to spatial data are used in different spatial OLAP prototypes, such as in [78, 179, 186]. With the goal of guiding development of spatial OLAP systems, Rivest et al. [179] define features that spatial OLAP tools should include in order to explore their potential. They categorize them in different groups, such as data visualization, data exploration, and data structures. Based on these features, they developed a prototype of spatial OLAP tools, which currently is commercialized under the name "JMap" [99]. Shekhar et al. [186] develop
3.9. SUMMARY

A map cube operator extending the concepts of data cube and aggregation to spatial data. Further, based on the classification used for non-spatial data, the authors present a classification and examples of different types of spatial measures, for example, spatial distributive, algebraic, and holistic functions. Han et al. [78] present GeoMiner, a spatial data mining system for online spatial data analysis. This system includes spatial as well as tabular and histogram presentations of data.

The extension of OLAP systems to include spatial data is also a concern of commercial software companies, where companies traditionally involved in business intelligence and in managing spatial data are joining their efforts for managing spatial and non-spatial data [95]. For example, OLAP company Business Objects offers an interface between ESRI's ArcView GIS and the Business Query tool allowing this way multidimensional reporting and analysis functionality, which results can be brought into GIS system for further geographic analysis. Other company Dimensional Insight, Inc, allows to integrate the MapInfo files into analytical tool; SAP implements mapping technology from ESRI to its Business Information Warehouse enabling interaction through a map-based interface; Sybase Industry Warehouse Solutions developed warehouse geocoded data models for industries like utilities and telecommunications [95].

Other works in spatial DWs relate to new index structures for improving performance (e.g., [136, 156, 175]), materialization of aggregated measures to manage high volumes of spatial data (e.g., [155, 193]), extension of spatial query languages for querying spatial multidimensional data (e.g., [167]), implementation issues for spatial OLAP (e.g., [78, 179, 186]).

3.9 Summary

In this chapter, we presented different elements to be included in a spatial multidimensional model, such as spatial levels, spatial hierarchies, spatial fact relationships, and spatial measures.

First, we referred to conceptual and object-relational representations for spatial levels. We presented different cases when these levels may have or not spatial attributes used for description purposes. We also referred to the case of thematic levels having spatial attributes.

For spatial hierarchies, we extended the classification of the different kinds of hierarchies proposed in Section 2.4 by the inclusion of spatial levels. Hierarchies may be fully or partly spatial depending on whether all their levels are spatial. For hierarchies having consecutive spatial levels, we emphasized that the summarizability problem may also occur due to the different topological relationships existing between hierarchy levels. We classified these relationships according to the complexity required for developing procedures for measure aggregation. Next, we presented the mapping of hierarchies to the object-relational model. Further, to ensure the semantic equivalence between the conceptual and the logical schemas during the transformation process, integrity constraints were exemplified mainly using triggers.

Finally, we presented the concepts of spatial fact relationships and spatial measures. For the former, we proposed the extension of the usual join operation for non-spatial fact relationships by inclusion of spatial join based on spatial predicates when more
than one spatial dimension is represented in a multidimensional model. Further, we referred to spatial measures and presented two cases: (1) when a measure is represented by a geometry, and (2) when a spatial measure is the result of a calculation using spatial or topological operators. We discussed the necessity of having spatial aggregation functions defined for measures when hierarchies are presented in the model. Moreover, we relaxed the requirement of having a spatial dimension to represent a spatial measure. Using Oracle 10g Spatial we presented the mappings and discussed implementation considerations for representing semantics of spatial fact relationships. We also showed the examples of different spatial functions including spatial aggregations functions useful for spatial DW applications.

Proposing spatial extensions for multidimensional models, we provide a concise and organized data representation for Spatial DW applications [12] and facilitate the delivery of data for spatial OLAP systems. Further, since it is platform independent, a conceptual multidimensional model allows to establish a communication bridge between users and designers. It reduces the difficulties of modeling spatial applications, since decision-making users do not usually possess the expertise required by software currently used for managing spatial data. Additionally, this conceptual model and the classification of topological relationships between hierarchy levels according to the required procedure for measure aggregation help implementers of spatial OLAP tools to have a common vision for spatial data in a multidimensional model and to develop correct and efficient solutions for spatial data manipulations.

The proposed mappings to the object-relational model along with the examples using a commercial system Oracle 10g Spatial, show the applicability of the proposed solutions in real-world situations and the feasibility of implementing spatial DWs in current commercial DBMSs. Further, integrated architectures, where spatial and thematic data is defined within the same DBMS, facilitate the system management simplifying data definition and manipulation. However, even though the mapping to the logical level is based on well-known rules, it does not completely represent the semantics expressed in the conceptual level. Therefore, additional programming effort is required to ensure the equivalence between conceptual and logical schemas.

Since the MultiDimER model is independent from any implementation, other systems could have been used in the place of Oracle. However, the presented implementation details would be different, as different systems provide different object-relational features and spatial extensions. The proposed mapping may also vary according to the expected usage patterns, for example, data mining algorithms.
Chapter 4

A conceptual model for temporal data warehouses

Current DW and OLAP models include an omnipresent time dimension that, as the other dimensions, is used for grouping purposes (the roll-up operation) or in a predicate role (the slice-and-dice operations). Nevertheless, even though the time dimension additionally serves as a time-varying indicator for measures, for example, sales in March 2005 in Figure 2.1, it cannot be used for representing the time when changes in other dimensions have occurred, for example, when a product changes its ingredients or when an employee changes his address. Therefore, usual multidimensional models are not symmetric in the way of representing changes. Consequently, the features of "time-variant" and "non-volatility" included in the definition of DWs (Section 2.1) only apply for measures leaving to applications the representation of changes occurring in dimensions.

On the other hand, as temporal databases (DBs) allow to manage historical data without any problems in representing changes to data and time when they occurred, several works (e.g., [1, 43, 133, 161]) consider that to solve problems in managing time-varying dimension data, DWs should include the same temporal support as temporal DBs. However, we believe that the proposal of a temporal extension for DWs cannot be done in a such "automatic" manner, therefore, in this chapter we present the temporal extension for DWs considering:

- Semantic differences between temporal DBs and DWs.
- Lack of conceptual models for temporal DWs that can be used as a communication bridge between users and designers.
- Lack of logical representations of temporal DWs in current DBMSs which could help implementers and also assess the usability of the conceptual models giving their logical representations.
- Inadequacy of relational DBs for representing complex data even though this kind of DBs are commonly used for mapping temporal DBs.

First, we refer to the inclusion of different temporal types into DWs. This temporal support is included for levels, links between levels forming a hierarchy, and measures.
For levels, we discuss temporal support for attributes and for a level as a whole. For hierarchies, we present different cases considering whether temporal changes to levels, to links between them, or to both, levels and links, are important to be kept.

Since source systems and DWs may have different time granularities, for example, source data is introduced on a daily basis yet DW data is aggregated by month, we consider two different situations: when measures are not aggregated before integration into a temporal DW and when these aggregations are realized. For the former, by the means of the real-world examples we show the usefulness of having different temporal types. For the latter, we discuss issues related to different time and data granularities and propose the inclusion of temporal types meaningful for aggregated measures.

Additionally, we present a mapping of a conceptual model for time-varying multi-dimensional data into a classical ER and into an object-relational (OR) models.

This chapter is organized as follows: Section 4.1 introduces concepts related to temporal databases. Section 4.2 describes motivation for using mapping into the ER and object-relational models. In Section 4.3 we propose the inclusion of different temporal types in temporal DWs. The definition of the temporally-extended MultiDimER model is given in Section 4.4. Section 4.5 includes mapping of different temporal types to the conventional ER model and to an object-relational model. Sections 4.6, 4.7, and 4.8 refer to our proposal for conceptual and logical representations for time-varying levels, hierarchies, and measures, respectively. Section 4.9 gives a metamodel for the temporally-extended MultiDimER model while Section 4.10 summarizes mapping rules from our model to the object-relational model. Section 4.11 surveys works related to temporal data warehouses and some aspects of temporal databases important for our model. The conclusions are given in Section 4.12.

4.1 Temporal databases: General concepts

The definitions presented in this section are based on [15, 73, 91, 93, 189, 190].

Temporal DBs, called also time-oriented databases or historical databases, are "databases that supports some aspect of time, not counting user-defined time" [91]. Time can be viewed in different ways, such as using so-called continuous, dense, or discrete representations. In continuous time model, time is isomorphic to real numbers, i.e., each real number corresponds to some time point; in the dense model the time is viewed as isomorphic to rational numbers, and in the discrete model time is isomorphic to integer numbers [189].

However, for database modeling it is not important which type of model is applied due to the fact that the important aspect is to have time values. The vast majority of research in temporal DBs (e.g., [53, 64, 93]) as well as this work assume a discrete time domain, where each integer number corresponds to a smallest time unit that system is able to represent called chronon. Depending on application requirements consecutive chronons can be grouped into a larger unit called a granule, such as a second, a minute, or a day. Granularity represents the size of the granule, i.e., it is the time unit used for specifying the duration of the granule. For example, if the granularity is one minute, then granule represents a specific minute.
4.1. TEMPORAL DATABASES: GENERAL CONCEPTS

Temporal data types  The concept of time can be applied for defining different temporal data types. The basic types are instants, periods, and intervals.

An instant is a time point on an underlying time axis. It has no duration and can be smaller than granule, however, due to the fact that the granule is the smallest unit representing time for an application, instant has to be "accommodated" into it. For example, if the granule is day, and the instant is specified in hours of this day, this instant will be defined as a day. The specific time value of an instant is called timestamp. The instant can have a special value such as forever or now [189, 219]. Both are used for indicating current time.

An instant can define time when some event occurs, i.e., the time when something "happens", for example, a child's birth in Figure 4.1 a). A set of instants can also be used for indicating time of an event, for example, en event such as earthquake can occur in different time instants.

Interval or period is the time between two instants, i.e., between a starting and an ending instants. It is represented using either a non-anchored (for example, two weeks) or an anchored (for example, [02/11/2004, 05/01/2005]) length of time for an interval or a period, respectively. Interval or period is used to represent state, i.e., something that has extent over time, for example, duration of a project as shown for the project A in Figure 4.1 b). Further, a set of intervals or periods can also be used for defining a state as shown for the project B in Figure 4.1 a).

Temporal types  The concepts presented above are required for representing different temporal types. Valid time (VT) specifies the time when the data is true in the modeled reality. Even though any data in the database may be associated with a valid time, it is usually supplied by the user and it may or may not be captured explicitly in the database. Valid time can be used for indicating the validity of an event or a state, therefore, an instant (or a set of instants) or a period (or a set of periods) are used. For example, the time when the specific salary was paid for an employee could be specified using a period.

1Usually called time dimensions; however, we use the term "dimension" in the multidimensional context.
Chapter 4. Temporal Data Warehouses

Transaction time (TT) indicates the time when the data is current in the database and may be retrieved. Transaction time is generally not a time instant, but have duration. Every period begins at the time when the data is inserted or modified and ends when the data is deleted or updated. This time is generated by the database system.

Both temporal types, i.e., valid time and transaction time, can be combined defining bitemporal time (BT). It indicates when the data is true in reality and when it is current in the database.

Further, in some applications, changes in time can be defined for an object as a whole, instead of using validity of its components (i.e., attributes), i.e., recording lifespan (LS) or existence time of an object, for example, the period of time when an employee was working for a company. Therefore lifespan can be seen as a valid time for a whole object. Notice that an object can also have assigned transaction time indicating the time when the database object is current in the database. The LT abbreviation is used when an object includes both, lifespan and transaction time support.

Conceptual and logical models Different conceptual and logical models have been proposed to represent temporal support. The conceptual models usually extend the already existing models, for example, the ER model. The proposed extensions either change the semantics of the existing constructs or introduce new constructs to the model [65]. Further, logical-level design is achieved by either mapping temporally-extended ER models to relational DBs (e.g., [66, 190]) or by using temporal normal forms (e.g., [93, 209]). However, there is still not a well-accepted procedure for mapping a temporal conceptual model, neither for achieving logical level design using temporal normal forms. Further, the relational representation produces a significant number of tables loosing the semantics of the modeled reality and incurring in performance problems due to the required join operations.

4.2 Motivation

In this section we present our rationale for transforming conceptual schemas designed using the MultiDimER model into schemas represented in a classical ER and object-relational models.

We choose the ER model since it is a well-known and widely-used model for conceptual modeling. Therefore, the ER representation of the constructs of the MultiDimER model allows a better understanding of their semantics. Further, the transformation of the ER model into operational data models is well understood (e.g., [48]) and this translation can be done using usual CASE tools.

On the other hand, in order to better assist the implementers who use the MultiDimER model for conceptual design of temporal DWs and who do not want to employ another conceptual model, we propose mapping rules that allow a direct translation of schemas from our model into schemas of to the object-relational model. We choose mapping instead of normalization (e.g., [93]) for several reasons. First, there are no well-accepted normal forms for TDBs even though some formal approaches exist, e.g., [93, 209, 211]. Further, the purpose of normalization is to avoid the problems of data
redundancy, potential inconsistency, and update anomalies. However, the usual practice in DWs is to de-normalize relations to improve performance and to avoid the costly process of joining tables in the presence of high volumes of data. This de-normalization can be done safely because data in temporal DWs is integrated from transactional or operational databases, which are usually normalized, and thus, there is no danger in incurring the mentioned problems. Finally, using a normalization approach may introduce a number of artificial relations that do not correspond to real-world entities, making the system more complex for designing, implementing, and querying.

We decided to use an object-relational model since it allows to better represent the real world grouping related data into a single row as was explained in Section 3.2.1. It also offers upward compatibility with the existing relational model allowing to "flatten" non-atomic data to a conventional first normal form (1NF). We show some examples of this logical representation using the SQL:2003 standard and its implementation in Oracle 10g.

4.3 Temporal types in DWs

The inclusion of different temporal types in temporal DWs is based on the semantic differences between the latter and temporal DBs:

- DW data is neither modified nor deleted. In contrast, in temporal DBs users change data directly usually recording the time of these changes as transaction time. Thus, the transaction time generated in a temporal DW plays a different role from the transaction time used in a temporal DB.

- Data in DWs is integrated from existing source systems whereas data in temporal DBs is inserted by users since it represents operational or transactional databases. Therefore, different temporal support may exist according to the types of source systems that integrate data into a temporal DW. Currently, this support is ignored in the DW context.

- DWs are designed according to analysis needs of decision-making users based on a multidimensional model with clearly distinguished measures and dimensions. The last two play different roles: measures are aggregated while dimensions are used to explore measures according to different criteria. On the other hand, design for temporal DBs is concerned with transactional or operational applications where all data is handled in a similar manner. Therefore, temporal support for DWs should consider differences in managing time-varying measures and dimensions.

Currently, DWs do not include different temporal support, even though they may be important for different reasons. The inclusion of lifespan in temporal DWs gives the possibility to perform analysis related to discover, for example, how sales change after the exclusion of some products. This support as well as valid time support also allow correct measure aggregations. For example, Eder et al. [43] include timestamps for dimension data considering valid time. Using this support, they propose mechanism

\footnote{We ignore modifications due to errors during data loading and deletion for purging DW data.}
for correct measure aggregations during roll-up operations allowing to span multiple periods of time.

Transaction time coming from source systems plays also an important role in temporal DW when traceability applications, for example, for fraud detection, are required. However, current works in temporal DWs handle this temporal support in different ways:

1. Ignore transaction time [19, 133].

2. Transform transaction time from source systems to represent valid time [127].

3. Consider transaction time generated in a temporal DW in the same way as transaction time is used in temporal DBs [127, 104], i.e., allowing to know when data was inserted, modified, or deleted from DWs.

However, using the first approach traceability applications cannot be implemented. The second approach is semantically incorrect because data may be included in databases after their period of validity has expired, for example, client's previous address. In the third approach, since data in temporal DWs is neither modified nor deleted, transaction time generated in a temporal DW represents indeed the time when data was loaded into a DW. The latter is called in the MultiDimER model data warehouse loading time (DWLT).

DWLT indicates when data is current in DWs and it can differ from transaction time or valid time of source systems due to the delay between the time when the changes have occurred in source systems and the time when these changes are integrated into a temporal DW. DWLT is important especially in active DWs [26] and in creating temporal DWs from non-temporal sources [215].

In the modeling process, application requirements determine the type of temporal support (none, VT, TT, BT, DWLT) that needs to be captured in each element of a temporal DW (attributes, levels, hierarchies, and/or measures). Obviously that depends on whether or not the different data sources of the temporal DW provide temporal support. These sources may also contain user-defined time attributes playing the role of valid time. Therefore, based on the classification of source systems given by Jarke et al. [89] and additionally considering temporal DBs as another kind of a source system similar to Abelló and Martín [1], in Table 4.1 we present the temporal support that can be obtained from source systems while building temporal DWs. The temporal type between parentheses indicates its possible existence in the system.

### 4.4 Temporal extension for the MultiDimER model

In this section we briefly present the temporal extension of the MultiDimER model. The formal definition of the model can be found in Appendix C.

The temporally-extended MultiDimER model may contain temporal as well as non-temporal elements. Therefore, the definitions of a schema, a level, a hierarchy, and a fact relationship remain the same as the ones presented in Section 2.2. Additionally,
### Table 4.1: Temporal types in source systems that may be used in temporal DWs.

<table>
<thead>
<tr>
<th>Type of source system</th>
<th>Description</th>
<th>Temporal types in sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Snapshot</strong></td>
<td>The access to the data in source systems is done through dump of its data. To find the changes, the current data is compared with the previous snapshot(s). The time when snapshot is realized determines neither transaction time nor valid time. Further, VT may be included as a user-defined attribute.</td>
<td>(VT)</td>
</tr>
<tr>
<td><strong>Queryable</strong></td>
<td>The sources offer a query interface. The detection of changes is done by periodic polling of the data items of the data source and by comparing them with the previous version. They may be considered as snapshot systems with the difference of having direct access to data.</td>
<td>(VT)</td>
</tr>
<tr>
<td><strong>Logged</strong></td>
<td>All actions are registered. The periodic polling of data is required for discovering what kinds of changes to which data are applied. The log files contain transaction time that can be retrieved. Similar to the previous kind of systems, valid time may also be a part of the system.</td>
<td>TT,(VT)</td>
</tr>
<tr>
<td><strong>Specific</strong></td>
<td>Each data is a particular case. There is not a general method for data extraction and detection of data changes. These systems can be considered as logged systems if either delta files or timestamps for attributes are available; otherwise, they may be treated as snapshot sources.</td>
<td>(TT),(VT)</td>
</tr>
<tr>
<td><strong>Callback and internal actions</strong></td>
<td>Sources that provide trigger, active capabilities, or programming environment so they are able to automatically detect changes of interest and notify those changes to the interested parties, i.e., to a temporal DW. The time and types of changes are detected without any delay.</td>
<td>TT,(VT)</td>
</tr>
<tr>
<td><strong>Replicated</strong></td>
<td>The detection of changes is done by analyzing the messages sent by the replication system. This may happen manually, periodically, or using specific criteria. Depending on features of the change monitor, this kind of systems can be considered as snapshot or callback systems.</td>
<td>(TT),(VT)</td>
</tr>
<tr>
<td><strong>Bitemporal</strong></td>
<td>Temporal DBs include the information that allows to know when the objects are valid in reality and when they are current in a DB. Since bitemporal aspect is already represented in source systems, transaction time and valid time are available.</td>
<td>TT,VT</td>
</tr>
</tbody>
</table>
we define a **temporal level** as a level, for which the application needs to keep its time-varying characteristics. A level may include temporal support independently of the fact that it has temporal attributes.

Temporal support is captured by the inclusion of different temporal types. Considering the arguments presented in Section 4.3 and further, based on the analysis of real-world applications\(^3\), we propose to include in our model lifespan (LS), valid time (VT), transaction time (TT), and bitemporal time (BT) coming from source systems (if available) and additionally, data warehouse loading time (DWLT) generated in a DW. We also allow the combination of lifespan (LS) and transaction time (TT) representing it as LT.

As was already defined in Section 2.2, the relationship between two levels is characterized by *cardinalities*, which restrict the minimum and the maximum number of members in one level that can be related to a member in another level. Since this relationship may include temporal support, the cardinality may be interpreted in two possible ways. The *snapshot cardinality* is valid every time instant whereas the *lifespan cardinality* is valid over the entire members lifespan. The former is represented as a continuous line and the latter as a dotted line with LS symbol. The presence of only one cardinality symbol indicates that both cardinalities are the same. The relationship between levels may include different temporal types: VT, TT, BT, and/or DWLT. There is no LS support for relationships since they do not exist by themselves without their participating levels. A hierarchy (resp. dimension) is temporal if it has at least one temporal level (resp. hierarchy) or temporal relationship between levels.

In the MultiDimER model measures are **temporal**, i.e., they always require to include a temporal type (VT, TT, BT, and/or DWLT). In this way, the usual time dimension is not required for indicating valid time for measures.

An example of using our notation is given in Figure 4.2\(^4\). It represents a schema for analysis of product sales where, for example, changes to products, categories, and relationships between them are kept. Several levels in the figure, such as Product, Category, include lifespan support (LS) that allows to track changes to a member as a whole, for example, inserting or deleting a product. To keep history of changes of the specific attributes, we use the symbol of corresponding temporal types, such as valid time (VT) for Size and Distributor in the Product level. Notice that not all elements must include temporal support: the Customer dimension in the figure does not have it.

The relationships between hierarchy levels may include temporal support. For example, valid time support for the relationship between the Product and Category levels allows to keep the history of assignments of products to categories. A many-to-one snapshot cardinality (a continuous line) indicates that product may belong only to one category in every time instant while a many-to-many lifespan cardinality (a dotted line with LS symbol) means that products may belong to many categories over its lifespan. Further, changes in measure values are represented using a temporal type (VT in the figure) instead of relying on the conventional Time dimension.

We refer in more detail throughout this chapter to the all described-above temporal

---

\(^3\)In Section 4.8 we will see more details.

\(^4\)More detailed description of our notation is given in Section D.5.
4.5 Mapping of temporal types

The temporal support in the MultiDimER model is added in an implicit manner, i.e., the timestamp attributes used for capturing a temporal aspect are represented using pictograms. Therefore, the transformation of the time-related data into classical non-temporal structures of the ER model requires additional attributes for timestamps, which are manipulated as usual attributes.

Mapping of temporal types to the ER model depends on whether these types are used for representing events or states. The former require an instant or a set of instants and the latter need a period or a set of periods. Notice that a set of periods or instants are used when the attribute has the same value in different periods or instants of time, for example, an employee working in the same section during different periods of time. Different options can be used for mapping valid time (VT) to the ER model:

1. A monovalued attribute for an instant.
2. A multivalued attribute for a set of instants.
3. A simple composite attribute for a period.
4. A multivalued composite attribute for a set of periods.
As lifespan can be represented by a period or a set of periods, it is transformed into a simple or multivalued composite attribute, respectively. The latter allows to include discontinuous lifespan, for example, a professor leaving for sabbatical during some period of time. For representing transaction time, the usual practice in temporal DBs is to use a period (or a set of periods) similar to lifespan. Since data warehouse loading time represents the time when data was loaded into a temporal DW, an instant is used for representing this temporal type; it is transformed into a simple attribute in the ER model.

To specify the mapping rules to the object-relational model, we use the different elements included in the SQL:2003 standard. SQL:2003 allows array and multiset to represent collections. The array type allows to store in a column variable-sized vectors of values of the same type while the multiset type allows to store unordered collections of values. Unlike arrays, multisets have no declared maximum cardinality. Composite types can be combined allowing nested collections, although this is considered an "advanced feature" in the standard.

SQL:2003 supports also structured user-defined types, which are analogous to class declarations in object languages. Structured types may have attributes, which can be of any SQL type including other structured types at any nesting. Structured types can be used as domain of a column of a table, as domain of an attribute of another type, or as a domain of a table. These structured types allow to group semantically related attributes.

The mapping rules to the object-relational model consider a multivalued attribute in the ER model as a multiset attribute while a composite attribute in the ER model as an attribute of a structured type comprising specified component attributes.

The mapping of different temporal types from our model to the object-relational model is based on the following rules:

**Rule 1:** A temporal type representing an instant is mapped to an attribute of date or timestamp type.

**Rule 2:** A temporal type representing a set of instants is mapped to a multiset attribute of date or timestamp type.

**Rule 3:** A temporal type representing a period is mapped to an attribute of a structured type composed of two attributes of date or timestamp type.

**Rule 4:** A temporal type representing a set of periods is mapped to a multiset attribute of a structured type consisting of two attributes of date or timestamp type.

For example, different options for valid time as explained at the beginning of this section can be represented in SQL:2003 as follows:

```sql
create type InstantType as date;
create type InstantSetType as (InstantType multiset);
create type PeriodType as (Pbegin date, Pend date);
create type PeriodSetType as (PeriodType multiset);
```

An example of a commercial object-oriented DBMS we use Oracle 10g [149]. Similar to the SQL:2003 standard, Oracle includes constructs that allow to represent collections. A varying array allows to store an ordered set of elements in a single row while a table type allows to have unordered set and to create nested tables, i.e., a table within
a table. The former corresponds to the array type of SQL:2003 and the latter to multiset. Further, as specified in Section 3.2.1 Oracle gives users the possibility to define object types similar to structured user-defined types in SQL:2003. The above specified declarations using SQL:2003 can be expressed in Oracle 10g as follows:

```
create type InstantType as object (Instant date);
create type InstantSetType as table of InstantType;
create type PeriodType as object (Pbegin date, Pend date);
create type PeriodSetType as table of PeriodType;
```

In the following, we present the temporal extension of the MultiDimER model applied to levels, hierarchies, and measures. For levels and hierarchies we give examples using valid time. Nevertheless, the results may be straightforwardly generalized for transaction time. We first refer to the conceptual representation and then, propose the mapping into the ER model and into the object-relational models.

4.6 Temporal levels

4.6.1 Conceptual representation

Changes in a level can occur either for attribute values (for example, changing a product’s size) or for a member as a whole (for example, inserting or deleting a product). Representing these changes in temporal DWs is important for analysis purposes, for example, to discover how the changes to the product’s size or the exclusions of some products influence sales.

To indicate the time when attribute values change, we use attribute timestamping since it better represents reality keeping changes only for the specified attributes. Figure 4.3 a) shows an example of a Product level that includes temporal attributes Size and Distributor. We group temporal attributes firstly, to ensure that both kinds of attributes (temporal and non-temporal) can be clearly represented and secondly, to include a smaller number of symbols. For indicating the specific temporal types we use the abbreviations VT, TT, BT, and DWLT.

Further, to indicate the time when a member exists in the modeled reality, i.e., its lifespan, we use the LS symbol next to the level name. The lifespan support for level members (Figure 4.3 b) indicates that each member includes the LS together with one value per attribute for non-temporal attributes and history of changes for temporal attributes. For example in Figure 4.3 b), every product includes the LS together with one value per non-temporal attribute (for example, Name) and the history of values for Size and Distributor.

Existing temporal models impose constraints for timestamped attributes and their corresponding object (entity) types; for example, the valid time of attribute values must be within the lifespan of the object (entity). Our model does not force it. In this way, different situations can be modeled, for example, a product that does not belong to a store inventory (it is not included in the master file), but it is on sales for defining its acceptance level. For this product, the valid time of temporal attributes may not be within the product lifespan. On the other hand, temporal integrity constraints may
be explicitly defined, if required, using a calculus that includes Allen’s operators [7].

Further, the lifespan as well as valid time used for attributes can be combined with transaction time or data warehouse loading time. In this way, users can obtain the information when the level member or the specific attributes values is current in a source or in a temporal DW, respectively.

### 4.6.2 Mapping of temporal levels

#### 4.6.2.1 Temporal attributes of a level

The transformation of the Product level from Figure 4.3 a) to the ER model is shown in Figure 4.4 a). The Product level corresponds to an entity type in the ER model. Non-temporal attributes are represented as monovalued attributes. Each temporal attribute is represented in the ER model as a multivalued composite attribute that includes an attribute for the value and another attribute for a temporal type. Notice, that using a multivalued attribute allows to have different values (for example, sizes or distributors) of the attribute in different periods of time. The validity of attribute values is represented by a period, which is a typical practice for dimension data in temporal DWs [17, 43]. Two possible representations are included depending on whether the valid time of the attribute Size is represented by a period (Figure 4.4 a)) or a set of periods (Figure 4.4 c))

As can be seen in Figure 4.3 a) and 4.4 a), the MultiDimER model provides better conceptual representation of time-varying attributes than the ER model. It contains less elements, it clearly allows to distinguish which data changes should be kept, and it leaves outside of user’s concerns some more technical aspects such as multivalued or composite attributes.

Applying the traditional mapping to the relational model, for example, from [48], gives three tables: one with all monovalued attributes and one for each multivalued attribute. All tables include an attribute for product keys. This relational representation is not very intuitive for users and designers since attributes of a level are stored as separate tables. It also has well-known performance problems due to the required join operations, especially if levels form part of a hierarchy.

An object-relational representation allows to overcome these drawbacks. It preserves more semantics keeping together in a single table a level and its temporal attributes. Figure 4.4 b) and c) show object-relational schemas using tabular representations containing the member key and all its attributes represented together in
### 4.6. TEMPORAL LEVELS

<table>
<thead>
<tr>
<th>Product</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product number</strong></td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td>VT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VB</td>
</tr>
<tr>
<td></td>
<td>Distributor (1,n)</td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
<td><strong>Size</strong></td>
<td><strong>Value</strong></td>
</tr>
<tr>
<td></td>
<td>VT</td>
<td>VE</td>
</tr>
<tr>
<td>1</td>
<td>Q6876</td>
<td>10 05/2002 08/2002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 09/2002 07/2003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 08/2003 NOW</td>
</tr>
<tr>
<td>2</td>
<td>QD555</td>
<td>18 05/2002 NOW</td>
</tr>
</tbody>
</table>

**Figure 4.4:** A level with temporal attributes: a) and c) the ER schemas, and b) and d) the object-relational representations.

The same table. In this representation we use the symbol * for denoting collection. Two possible representations are included depending on whether the valid time of the attribute Size is represented by a period (Figure 4.4 b)) or a set of periods (Figure 4.4 d)). For simplicity we do not represent in the figure the Distributor attribute, which can be mapped similarly to the Size attribute. The object-relational representation corresponds to a temporally grouped data model [36], which is considered as more expressive for modeling complex data [211].

The mapping of temporal attributes from our model to the object-relational model is straightforward: **Rule 5:** An attribute with temporal support is mapped to the object-relational model as a multiset attribute of a structured type composed of two attributes: one for representing the value and another one for the associated temporal type.

For example, given the declarations for the temporal types in Section 4.5, the type for the Size attribute is defined as follows for Figure 4.4 b) and d), respectively:

```sql
create type SizeType as (Value real, VT PeriodType);
create type SizeType as (Value real, VT PeriodSetType);
```

Since Size is a multivalued attribute, we represent it as a collection type using either array or multiset, i.e., using one of the following:
create type SizeCollType as (SizeType multiset);
create type SizeCollType as (SizeType array(7));

Mapping a level to the object-relational model is straightforward once the types for its attributes are defined:

**Rule 6a:** A level is mapped to a relation containing all its attributes and an additional attribute for a key.

The Product level can be represented in SQL:2003 using two types of tables. *Relational tables* are usual tables, although the domains for attributes are all predefined or user-defined types. *Typed tables* are tables that use some structured types for their definition. In addition, typed tables contain a *self-referencing column* keeping the value that uniquely identifies each row. Such column may be primary key of the table, it could be derived from one or more attributes, or it could be a column whose values are automatically-generated by the DBMS, i.e., surrogates. Since the surrogates are important in DWs, we use a typed table for representing the Product level. The declaration of a typed table requires first the definition of a type (ProductType) for the elements of the table:

```sql
create type ProductType as (
    Number integer,
    Name character varying(25),
    Description character varying(255),
    Size SizeCollType) ref is system generated;
```

Then, a typed table (Product) is created:

```sql
create table Product of ProductType (
    constraint prodPK primary key (Number),
    ref is Sid system generated);
```

The clause "ref is Sid system generated" indicates that Sid is a surrogate attribute automatically generated by the system.

To define the Product table in Oracle 10g we slightly modify the definition of a temporal type represented by a set of periods (Section 4.5) using now a varying array instead of a table:

```sql
create type PeriodType as object (
    Pbegin date, Pend date);
create type PeriodSetArrType as varray(7) of PeriodType;
create type SizeType as object (
    Value number, VT PeriodSetArrType);
create type SizeTabType as table of SizeType;
create type ProductType as object (
    Number number(10),
    Name varchar2(25),
    Description varchar2(255),
    Size SizeTabType);
```

---

5In Section 2.3 we already justified the importance of having surrogates for DW implementation.
6For simplicity, in the examples we omit full specification of constraints and additional clauses required by the standard.
4.6. TEMPORAL LEVELS

create table Product of ProductType
    constraint prodPK primary key (Number)
    nested table Size store as SizeNT.TAB
    object identifier is system generated;

The definitions in Oracle to some extent differ from those in SQL:2003, however, they represent the main features included in the standard. For example as specified in Section 4.5 two different types of collections can be used, i.e., varying array and table types. Typed tables in SQL:2003 correspond to object tables in Oracle (e.g., the Product table) and they either can include automatically-generated surrogates (as in the previous example; the default option) or alternatively can use the primary key for that purpose.

However, it is important to consider the differences at the physical level when using different options that Oracle provides for representing collections. For example, varying arrays are in general stored “inline” in a row^ and cannot be indexed. On the other hand, rows in a nested table can have identifiers, can be indexed, and are not necessarily brought to memory when accessing the main table (if the field defined as a nested table is not specified in the query). Nested tables require to specify their physical locations. For example, as shown in the previous declaration, the physical location for the nested table Size must be explicitly defined when the Product table is created. Therefore, the choice between nested tables and varying arrays must be done according to application specificities, for example, which data is accessed, which operations are required, as well as taking into account performance issues.

Until now we have discussed the representation of valid time. However, this temporal type can be combined with transaction time and/or data warehouse loading time. The latter two can be mapped according to the explanations given in this section and in Section 4.5.

4.6.2.2 Levels lifespan

The ER representation for a level that includes lifespan support (Figure 4.3 b)) is shown in Figure 4.5 a). It is based on the explanation given in Section 4.5 for mapping lifespan temporal type to the ER model. Two component attributes in the figure, LB and LE indicate, respectively, the beginning and the ending instants of the lifespan.

Figure 4.5 b) shows the object-relational representation where the member surrogate, the lifespan, and all its attributes are represented together in the same table. On the contrary, representing the lifespan in the relational model would introduce an additional table with a foreign key referring to the surrogate attribute of a level and with two attributes representing LB and LE.

The SQL:2003 declaration for the Product type includes an additional attribute representing the temporal element of the lifespan as follows:

```sql
create type ProductType as (
    LS PeriodSetType,
    Number integer,
    Name character varying(25),
);```

^If they are less than 4000 bytes or not explicitly specified as a large object (LOB).
Figure 4.5: A level with lifespan: a) the ER schema and b) the object-relational representation.

Description character varying(255),
Sizes SizeCollType) ref is system generated;
When lifespan is combined with transaction time and data warehouse loading time, corresponding mapping should be applied (see Section 4.5).
We extend now Rule 6a indicating the mapping rule for lifespan support:
Rule 6b: If a level has LS support, in the object-relational model an additional attribute as specified by Rules 3 or 4 should be included.

4.7 Temporal hierarchies

4.7.1 Conceptual representation

The MultiDimER model allows to represent hierarchies that contain several related levels. Given two consecutive levels in a hierarchy, the levels, the relationship between them, or both, the levels and the relationship between them, may be temporal. We examine next different situations that may occur in temporal hierarchies.

4.7.1.1 Non-temporal relationships between temporal levels

Temporal levels can be associated with non-temporal relationships. Temporality in levels requires to keep changes for attributes or for lifespan of members. On the other hand, non-temporal relationships indicate that either these relationships never change or if they do, only the last modification is kept.

Nevertheless, if level members change the key attributes used for traversing from one level to another during the roll-up and drill-down operations, incorrect analysis scenario may occur. For example, suppose in Figure 4.6 that the Product and Category levels include valid time for key attributes. Let us suppose that Product A belongs to category C (Figure 4.7 a)) and this category is divided in two new categories called C1 and C2 at time instant t2. If the relationship product A – category C1 replaces a
4.7. TEMPORAL HIERARCHIES

Figure 4.6: An example of a non-temporal relationship between temporal levels.

previous version of product A - category C (Figure 4.7 b)), the analysis previous to this change (previous to the instant \( t_2 \) in Figure 4.7 b)) cannot be made since there is not link before time \( t_2 \) that leads from product A to a valid category. Therefore measure cannot be aggregated for any time instant less than \( t_2 \). Similar situation occurs when the relationship is not modified with the difference that the analysis cannot be made after instant \( t_2 \).

Figure 4.7: An example of incorrect analysis scenario: a) before and b) after splitting categories.

To avoid an incorrect management of hierarchies as described above, we allow temporal levels with non-temporal relationships between them only for those levels that do not keep their LS and do not include VT for their key attributes. For example in Figure 4.6 the only changes allowed are those that do not affect relationships between members of these levels, for example, a product changes its distributor but it belongs to the same category.

Notice that transaction time and data warehouse loading time can always be included for all levels or attributes since this temporal type only refers to the time when temporal DW members or attribute values are available in source systems and in a temporal DW, i.e., they do not affect validity of members and attributes.

4.7.1.2 Temporal relationships between non-temporal levels

To represent temporal support allowing changes in relationships between levels, we place the corresponding temporal symbol, for example, VT, on the link between hierarchy levels as can be see in Figure 4.8.

Temporal relationships allow to keep track of the evolution of links between parent and child members. Non-temporal levels indicate that either these levels never change or if they do, only the last modification is kept. However, if level members change the key attributes used for traversing from one level to another during the roll-up and drill-down operations, dangling references may occur. For example, suppose for the
example in Figure 4.8 that the employee E1 is assigned to section S1 (Figure 4.9 a)). At instant \( t_2 \) section S1 is divided in two new sections S11 and S12, overwriting the previous value of section S1 (Figure 4.9 b)). The employee E1 is assigned now to section S11. Since the relationship is temporal, both assignments with corresponding valid time will be kept, i.e., to the previous section S1 and to the current section S11 (Figure 4.9 b)). However, during the roll-up operations before the instant \( t_2 \), references to non-existing section S1 will be made.

To avoid dangling references and consistency during roll-up and drill-down operations, we allow non-temporal levels to be linked with temporal relationships only if the values of the level members do not change but the changes to relationship between levels are required to be kept.

Some examples of this changing relationships, called transitions [219] include evolution and extension. The former occurs when a child member ceases to be related to one parent member and is assigned to another one, for example, an employee is assigned to a new section (Figure 4.10 a). The latter takes place when a child member belongs to the original parent member and additionally a new relationship with a different parent member is included, for example, an employee is assigned to the new section, leaving him also in the old section (Figure 4.10 b).
4.7. TEMPORAL HIERARCHIES

4.7.1.3 Temporal relationships between temporal levels

Temporal levels may include lifespan support and/or temporal key attributes. This temporal support ensures to keep all changes occurring to level members and/or attribute values. However, these changes may affect the relationships with members of child and/or parent levels. For example, the geographical distribution in Europe during the last 20 years has changed since some countries cease to exist, are merged, or split [43]. As a consequence, in hierarchies representing this geographical distribution, reassignment of states or provinces to new countries may be required.

Therefore, in the case when levels include LS support and/or VT for key attributes, to avoid incorrect management of hierarchies as described in Section 4.7.1.1, relationships between temporal levels must also be temporal. This restriction in the Multi-DimER model, as in most models in temporal DBs, helps to avoid incorrect analysis scenarios and dangling references.

![Figure 4.11: An example of temporal relationships between temporal levels.](image)

In the example of Figure 4.11 a) all levels are temporal. They include LS for dimension levels and VT for some attributes. Further, the relationships between levels are also temporal. Suppose that the sales company is in an active development and changes to Sales districts may occur to improve the organizational structure. These changes may affect the relationship with members of the Store and/or State levels.

The constraint for temporal relationships between temporal levels forming a hierarchy is more restrictive than the one usually used for temporal relationships between temporal objects in temporal DBs. In the latter, the valid time of a relationship instance must be included in the intersection of the valid times of participating objects. In multidimensional hierarchies it is further required that every valid child (respectively parent) member must be associated with at least one valid parent (respectively child) member in order to ensure correctness of the roll-up and drill-down operations. Validity can refer to the LS of a level as well as to the VT of key attributes leading to the following constraints:

1. Every time point included in the LS of a level must be included in the LS of some member of the next level, i.e., a valid child member must have a valid parent member and vice versa. If this condition is not fulfilled, structural changes to hierarchies could occur, for example, forcing some level members to skip the current parent level.

2. Every time point included in the VT of a key attribute (i.e., used for aggregation...
purposes) of a child (respectively parent) member must be included in the VT of some key attribute of a parent (respectively child) member.

4.7.1.4 Conditions for including temporal support in hierarchies

Based on the explanations given in the previous sections, in this section we will summarize the conditions for including temporal support in multidimensional hierarchies. Further, we also include cases not seen until now when either temporal or non-temporal relationships exist between a temporal and a non-temporal levels.

1. Temporal levels and non-temporal relationships between them: temporal features can only be applied for attributes that do not participate in the roll-up and drill-down operations.

2. Temporal levels and temporal relationships between them: levels, attributes, and links indicating the relationship between levels can be temporal.

3. Non-temporal levels and temporal relationships between them: no modifications to level members are allowed.

4. Non-temporal levels and non-temporal relationships between them: changes to dimension data cannot be kept. This is the current situation where implementation "tricks" must be used to represent changes to level members and/or to relationships between them as described in Section 1.2.1.

5. One temporal and one non-temporal level and temporal relationships between them: similar to Case 3, thus members of a non-temporal level cannot be modified.

6. One temporal and one non-temporal level and non-temporal relationships between them: similar to Case 1, thus temporal level only can have temporal types for non-key attributes.

4.7.1.5 Snapshot and lifespan cardinalities

Cardinalities in a non-temporal model indicate the number of members in one level that can be related to member(s) in another level. In our model, this cardinality may be considered for every time instant (snapshot cardinality) or over members lifespan (lifespan cardinality).

The lifespan cardinality may be different from the snapshot cardinality because of the changes in hierarchies, both in levels and in relationships between them. Thus, when these temporal changes must be kept, both lifespan and snapshot cardinalities may be considered.

In the MultiDimER model the snapshot cardinality is by default equal to the lifespan cardinality; however, if these cardinalities are different, a dotted line with the LS symbol is inserted and it indicates the lifespan cardinality as shown in Figure 4.12.

In the example in Figure 4.12, the employee snapshot and lifespan cardinalities for the hierarchy Works are many-to-many indicating that an employee can work in more
Figure 4.12: Snapshot and lifespan cardinalities between hierarchy levels.

than one section at the same time instant and over his lifespan. On the other hand, the snapshot cardinality for the hierarchy Affiliated is one-to-many, and the lifespan cardinality is many-to-many indicating that in every time instant an employée can be affiliated only to one section, but over his lifespan he can be affiliated to many sections.

Further, the constraint imposed on the cardinalities requires the minimum value as well as the maximum value of the lifespan cardinalities to be equal or greater than minimum and maximum values of the snapshot cardinalities, respectively.

4.7.2 Mapping of child-parent relationships

Temporal or non-temporal levels can be linked with either temporal or non-temporal child-parent relationships. In Section 4.6.2 we already discussed the mapping procedures for the temporal levels and in Section 2.3 we specify the rules for mapping non-temporal levels. In this section we will present mapping of child-parent relationships considering two cases: when these relationships are temporal or not.

4.7.2.1 Mapping of non-temporal relationships

Non-temporal relationships are the usual binary relationships in the ER model as defined in Section 2.2. Figure 4.13 a) represents the transformation to the ER schema of the MultiDimER schema in Figure 4.6®.

For obtaining the corresponding object-relational schema, first we represent each level as explained in Section 4.6.2. Then, we use the traditional mapping for binary many-to-one relationships and include a parent key (for example, Category in Figure 4.13) in the child level table (for example, Product in Figure 4.13). Since the levels are identified by surrogates, which are time-invariant, this mapping does not depend on whether the levels are temporal or not.

Rule 7: A non-temporal many-to-one relationship between child and parent levels is mapped to the OR representation by including a parent key in the child level table.

For example, the mapping of the Product level and the Product-Category relationship gives the relation shown in Figure 4.13 b).

To define the Product table in SQL:2003 first we need to create a typed table Category with the surrogate in the Sid attribute:

```
create type CategoryType as (  
    Name character varying(25),  
    ... ) ref is system generated;
```

®For simplicity we only present one temporal attribute.
create table Category of CategoryType (  
... /* some constraints *//,  
ref is Sid system generated);  

Then, we define a ProductType and a Product table as follow:  
create type ProductType as (  
Number integer,  
...  
Sizes SizeCollType,  
CategoryRef ref(CategoryType) scope Category  
references are checked) ref is system generated;  
create table Product of ProductType (  
constraint prodPK primary key (Number),  
ref is Sid system generated);  

Notice that the ProductType includes a reference (ref) type that points to the corresponding row in the Category table.  

The Oracle declarations for representing the Product-Category hierarchy are very similar to those of SQL:2003 since reference attributes are also provided in Oracle:  
create type CategoryType as object (  
Name varchar(25),  
...);  
create table Category of CategoryType (  
/* some constraints */);
create type ProductType as object (  
    Number number(10),  
    ...  
    Sizes Size.CT,  
    CategoryRef ref(CategoryType));  
create table Product of ProductType (  
    constraint prodPK primary key (ProdNumber),  
    constraint prodFK foreign key CategoryRef references Category);

4.7.2.2 Mapping of temporal relationships

Temporal relationships can link either non-temporal levels (Figure 4.8 and 4.12) or temporal levels (Figure 4.11). Different options for mapping temporal relationships exist depending on the maximum cardinality of the child level. Notice that in order to have meaningful hierarchies, we suppose that the maximum cardinality of the parent level is always N.

**Maximum child cardinality equal to 1** In this case, the snapshot and lifespan cardinalities are the same (i.e., many-to-one) allowing a child member belongs to at most one parent member during its entire lifespan. For example, in Figure 4.14 a) an employee may work only in one section and if he returns after a leave, he must be assigned to the same section. Figure 4.14 b) shows the corresponding ER model where valid time is represented as a multivalued composite attribute since an employee can be hired several times for the same section.

For the object-relational representation we can either create a separate table for the Works relationship (Figure 4.14 c) or include a multivalued attribute in the child-level table, for example, include in the Employee table an attribute for the Section surrogates with its temporal characteristics (Figure 4.14 d).

The definition of the Works relation in SQL:2003 requires that Employee and Section tables have already been declared as typed tables. We do not use a typed table for representing the Works relationship since the relationship does not exist without their levels.

create type WorksType as (  
    SectionRef ref(SectionType) scope Section  
    references are checked,  
    VT PeriodSetType);  
create table Works (  
    EmployeeRef ref(EmployeeType) scope Employee  
    references are checked,  
    InSection WorksType);  

The SQL:2003 declaration for Employee in Figure 4.14 d) requires to include the In-Section attribute of the WorksType in the Employee type. This representation expresses in a better way the semantics of the relationship since all different periods indicating when an employee works in the section are included in the same row.

For simplicity we do not present levels attributes.
Maximum child cardinality equal to N  Two different cases fit in this category: (1) the snapshot and lifespan cardinalities are different (in Figure 4.15 a) continuous and dotted lines, respectively), i.e., a child member is related to one parent member at every time instant and to many parent members over its lifespan and (2) the snapshot and lifespan cardinalities are the same (Figure 4.15 b)), i.e., a child member is related to many parent members at every time instant\footnote{Called in \cite{116} non-strict hierarchies.}. Both cases are handled in the same way since for the first case when cardinalities are different, we consider the lifespan cardinality. The snapshot cardinality is represented as a constraint on the lifespan cardinality indicating that every employee can include only one NOW value for the timestamp indicating ending instant of a valid time.

The mapping to the ER model is straightforward. It can be represented in the same way as in Figure 4.14 b) except that the cardinalities are many-to-many.

As for the previous case (Figure 4.14), two different object-relational representations may be used: either a separate table for the Works relationship or a table for a child level (Employee) with an additional attribute representing this relationship; the latter is shown in Figure 4.15 c). Notice, that the foreign key is represented as set of values since an employee can work in many sections over his lifespan. This leads to the inclusion of a multiset type for the InSection attribute of the EmployeeType type in the
Figure 4.15: Temporal relationships linking non-temporal levels: a) and b) different versions of the MultiDimER schema and c) the object-relational representation for the Employee level.

SQL:2003 declaration:

```sql
create type EmployeeType as (
    EmplID integer,
    ...,
    InSection WorksType multiset);
```

Summarizing, for mapping temporal relationships between the child and parent levels to the object-relational model the following rule is used:

**Rule 8**: Firstly, a structured type composed of two attributes is defined: one attribute for representing surrogates of the parent level and another one for the corresponding temporal type. Secondly, this structured type is used for defining a simple or a multiset attribute depending on whether the cardinality between child and parent levels is many-to-one or many-to-many, respectively. Let us call this attribute TemRel. Finally, one of two possible object-relational representations can be used:

1. Creating a new relation that contains an attribute for the surrogate keys of the child level and the TemRel attribute.

2. Extending the relation corresponding to the child level with the TemRel attribute.

Even though the second option is more expressive, the choice among the alternative object-relational representations may depend on physical-level considerations for the
particular DBMS, such as join algorithms, indexing capabilities, etc. For example, defining the InSection attribute as a nested table in Oracle 10g, will require a join of two tables, thus not offering any advantage with respect to the solution of a separate table for the Works relationship. Notice that for the previous cases, the relational model only gives the option of creating a separate table for the Works relationship.

The ER and object-relational representations are the same if levels are temporal since the surrogates of child and parent members participating in a relationship are time-invariant. A simplifying example of a schema using the MultiDimER model and its corresponding object-relational representation are given in Figure 4.16 a) and b), respectively.

![Diagram](image)

Figure 4.16: Temporal relationship between temporal levels: a) the MultiDimER model and b) the object-relational representation.

Additionally, a relationship between levels can include transaction time and/or data warehouse loading time for which the mapping specified in Section 4.5 can be used.
4.8 Temporal measures

Currently, multidimensional models allow to represent changes to measures and the time when they occur using a time dimension. Since this dimension cannot be used for representing time of changes to dimension data, in the previous sections, we propose the inclusion of temporal support for levels and hierarchies. However, we consider that it is important to give more symmetry to the multidimensional model and provide similar temporal support for the whole DW, i.e., for levels, hierarchies, and measures.

A simplifying example with temporal support for the whole DW schema was given in Figure 4.2. In this example, changes in measure values are represented using a temporal type (VT in the figure). Similar to the attributes of a level, attribute timestamping is used since it allows to include specific temporal types for each measure. The presence of only one abbreviation allows to minimize the number of symbols.

The important question is whether it is necessary to have the time dimension in the model after including temporal types for measures. If the time dimension has only the attributes that contain a granule, this dimension is not required anymore [17]. The additional information, for example, if it is the week day, the last day of the month, can be obtained applying time manipulation functions. However, in some temporal DW applications this calculation can be very time-consuming or some data cannot be acquired at all, for example, occurred events\textsuperscript{12}. Thus, this dimension will or will not be included depending on users' requirements and the capabilities provided by DBMSs.

Temporal support for measures in current multidimensional models includes only valid time. Nevertheless, different from valid time support may be required for measures expanding analysis spectrum for decision-support processes.

In this section, we first present examples where different temporal types for measures are important for analysis purposes. To simplify the discussion, we divide it in two parts: (1) when time granularities attached to measures in source systems and in a temporal DW are the same, i.e., measures are not aggregated, and (2) when this granularity is finer in source systems, i.e., measures are aggregated with regard to time.

The last case when time granularity for measures in source systems is greater than in a temporal DW is meaningless since detailed data cannot be obtained from aggregated data without loss of information. We finish this section proposing the mapping of temporal measures to the ER and object-relational models.

4.8.1 Temporal support for non-aggregated measures

Considering that temporal support in temporal DWs depends on both the availability of temporal types in source systems and the kind of required analysis, we present next some examples that refer to these two aspects. Based on these examples, we then show that different types of temporal support for measures can enrich analysis. For simplicity, we use non-temporal dimensions; the inclusion of temporal dimensions is straightforward.

\textsuperscript{12}In Costa Rica when an event such as earthquake occurs, sales of water bottles and canned food increases.
Case 1. Sources: non-temporal, temporal DWs: DWLT  In real-world situations, many sources can be non-temporal or temporal support is implemented in an ad-hoc manner that can be both inefficient and difficult to automate. Since a temporal DW integrates data from different sources, even though they may have temporal support, the integration process, for example, checking the time consistency between different source systems, can be too costly. Nevertheless, decision-making users may require the history of how source data has evolved [215]. Thus, the measure values can be timestamped with data warehouse loading time indicating the time when this data is loaded into the warehouse.

In the example in Figure 4.17 users require the history of Product inventory considering different suppliers and warehouses. The DWLT abbreviation next to the measures indicates that measure values will be timestamped when loaded to the temporal DW.

![Figure 4.17: Inclusion of DWLT for measures.](image)

Case 2. Sources and temporal DWs: VT  This case occurs when source systems can offer valid time. This valid time is required in a temporal DW for representing events or states. Figure 4.18 a) gives an example of an event model used for the analysis of banking transactions and and Figure 4.18 b) shows a state model for analysis of history of employees' salaries.

Different types of queries can be formulated for each of these models. For example, in Figure 4.18 a) we can analyze clients’ behavior related to time between operations, a maximum or minimum withdraw, total amount for withdraw operations, total number of transactions during lunch hours, the frequency of using a specific ATM, etc. This model also allows to analyze clients’ sequential behavior to avoid for example, cancellation of an account in the bank or to promote some new services. Further, similar models can be used for different application domains, for example, for an insurance company, for a library, for a transportation.

On the other hand, the model in Figure 4.18 b) can be used for the analysis of the evolution of salary paid for employees according to different criteria, for example, changes in professional skills or changes in amount of salary paid for employees for participation in a specific project.
4.8. TEMPORAL MEASURES

Figure 4.18: Inclusion of VT for measures representing a) event and b) state models.

Case 3. Sources: TT, temporal DWs: VT In this case, users require to know either the time when an event occurred in reality or a period of validity for data representing state. However, source systems can only offer the time when data was modified in a source system, i.e., transaction time. Thus, the analysis if transaction time can be used for approximating valid time should be made. For example, if a measure represents clients' account balance, valid time for this measure can be calculated considering transaction times of two consecutive operations.

Nevertheless, as already explained before, transaction time cannot always be used for calculating valid time, since some data can be inserted in source systems (registering transaction time) when they are not valid in the modeled reality, for example, employee’s previous salary. In many applications, only the user knows valid time. Therefore, it is incorrect to assume that if valid time is not given, the data is considered valid while it is current in source systems [127].

The transformation from transaction time to approximate valid time must be a very careful decision, and the designer must make aware decision-making users about the imprecision that may be introduced. Some kinds of systems can give a better result (for example, transaction or real-time systems) while in others this transformation should be forbidden (for example, a database created currently about events occurred in the past).
Case 4. Sources: VT, temporal DWs: VT and DWLT In the previous two cases, we include valid time in a temporal DW, which is the most common practice. However, the addition of data warehouse loading time for measures can give the information since when the data has been available for the decision-making process.

<table>
<thead>
<tr>
<th>Sales</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>10</th>
<th>5</th>
<th>no sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 4.19: An example of the usefulness of having VT and DWLT.

The inclusion of data warehouse loading time can help to better understand decisions made in the past and to adjust loading frequencies. For example, based on a growing tendency of product sales during weeks 10, 11 and 12 (Figure 4.19), it was decided to buy more products. However, only in the next DW load, occurred eight weeks later, a new situation has been revealed: a sudden decrease of sales. Thus, an additional analysis can be performed to understand the causes of these changes in sales behavior. Further, the decision of more frequent loads may be taken.

Case 5. Sources: TT, temporal DWs: TT (DWLT, VT) DW data can be needed for traceability applications (for example, for fraud detection) where changes to data and time when they have occurred should be available. That is possible if source systems have transaction time, since with this temporal support past states of a database are kept.

Figure 4.20: An example of a temporal DW for insurance company.

An example given in Figure 4.20 is used for an insurance company having as an analysis focus the amount of insurance payments. Since investigators suspect an internal fraud by modification of the amount of insurance paid to clients, the detailed information is required indicating when changes in measure values have been introduced. Notice that including in addition data warehouse loading time would give the information since when data has been available for the investigation process. Further, the inclusion of valid time would allow to know when the payment was received by client. In many real systems, the combination of both, transaction time and valid time, i.e., bitemporal time will be included.
4.8. TEMPORAL MEASURES

Case 6. Sources: BT, temporal DWs: BT and DWLT Data in temporal DWs should provide a timely consistent representation of information [26]. Since some delay may occur between the time when the data is valid in the reality, when it is known in the source systems, and when it is stored in the DW, it is sometimes necessary to include valid time, transaction time as well as data warehouse loading time. Figure 4.21 shows an example of the usefulness of having these three temporal types. This example is based on the conceptual model for managing temporal consistency in active DWs [26]. The latter combines DWs with the mechanism of E–C–A (event–condition–action) rules used in active DBs [217].

Figure 4.21: An example of the usefulness of having VT, TT, and DWLT for temporal DWs.

In this example, a salary 100 with valid time from the second to fifth months was stored at the third month (TT) in a source system. Afterwards, at the eighth month (TT) a new salary was inserted with value 200 and valid time from the sixth month until NOW. When data was loaded into temporal DW at DWLT, the value of the salary was unknown. In the next loading DWLT the value 100 was stored in the temporal DW. However, depending on which instant of time users want to analyze different values can be retrieved, for example, the salary for the first month is unknown, but for the fourth month the value 100 is retrieved. For more details and analysis, readers can refer to [26]; they specify additional conditions to ensure timely correct states during analytical processing.

4.8.2 Temporal support for aggregated measures

In this section, we consider different time granularities between source and temporal DW systems. We will analyze how to match these different time granularities and also how to aggregate measures to which these time granules are attached. We also refer to temporal types that can be used for aggregated measures in temporal DWs.

Notice that loading frequencies in temporal DWs may be different from the time granularity used for measures, for example, data may be stored using as a granule month but the loading is performed every quarter. We suppose that data can be kept in source systems before loading into a temporal DW.

4.8.2.1 Different time granularities

Measures in temporal DWs can be aggregated with respect to time before loading, thus an adequate mapping between multiple time granularities of a source system and a temporal DW should be considered. Two mappings are distinguished: regular and irregular [42].
In the regular mapping some conversion constant exists, i.e., one granule is further partitioning of another granule, so if one granule is represented by an integer it can be converted to another by a simple multiply or divide strategy, for example, minutes and hours or days and weeks.

In the irregular mappings, granules cannot be converted by a simple multiply or divide, for example, month and days, since each month is formed by a different number of days. Other examples include granularities that include gaps [16], for example, business week that contains a five days separated with a two-day gap. Thus, mapping between different time granules must be specified explicitly. For example, Dyrsen [42] requires C functions with a detailed specification to obtain the desired conversion. Such mapping functions may be quite complicated. For example, at a university the statistics of using computer laboratory may be captured every day; however, a temporal DW may include these statistics using as a granule semester. Since academic semesters start and end at different days and months, and have a different number of days, the mapping function must be customized according to user specifications.

Some mappings between different granularities are not allowed in temporal DBs [37, 42], for example, between weeks and months since a week can belong to two months. Nevertheless, this situation can be found in DW applications, for example, the analysis of employees' salaries for each month having some employees with a salary received on weekly basis. We call the mapping of such granularities forced. It requires a special handling during measure aggregations, to which we refer in the next section.

4.8.2.2 Aggregation of measures with valid time

After considering the mapping between different time granularities, the aggregation of measure values must be realized taking into account (1) applied functions, for example, sum, average and (2) the type of measures [111], i.e., flow (e.g., monthly income), stock (e.g., inventory of product), or value-per-unit (e.g., item price) as explained in Section 2.1. For example, suppose that a source System records the precipitation level every minute while the granularity in a temporal DW is day. The aggregated measures could be daily average precipitation or daily minimum/maximum precipitation.

However, in some cases the procedures for measure aggregations could be more complex. A simplifying example is given in Figure 4.22 where the time granularity in a source (month) requires a regular mapping to the time granularity in a DW (quarter). This example includes different cases: (1) a period of time that the same salary is paid can overlap several quarters (salary 20 and 40), (2) during a quarter several periods may be included indicating different amounts of salary (quarter 2), and (3) during several months of a quarter an employee does not receive a salary (quarter 3). A user requires as a temporal DW measure average salary per quarter. For the first quarter, the average value is calculated easily. For the second quarter, the simple average does not work, thus the weighted mean value may be given instead. However, for the third quarter, a user should indicate how the value must be specified. In the example, we opt for giving an undefined value. Nevertheless, if instead of using average salary we use the sum (total salary earned during a quarter), the measure value for the quarter 3 can be defined.

Real situations could be more complicated demanding clear specifications of coer-
4.8. TEMPORAL MEASURES

Figure 4.22: Example of coercion function for salary.

con functions [134] or semantic assumptions [16]. The idea of coercion functions is not new in the temporal DB research community. For example, Merlo et al. [134] use them for calculating values attached to timestamps of different granularities between subtypes and supertypes, Wang et al. [209] for defining lossless decompositions necessary for temporal normal forms, and Bettini et al. [16] for proposing a new framework for temporal DBs that facilitates specifications of users' queries. In the previous example in Figure 4.22, we use a user-defined coercion function [134] that states if a temporal DW granule is not covered in its totality by valid time of one or several salaries, the average salary is undefined.

Further, other solutions already known in temporal DBs may also be applied. We give an example inspired from [16] transforming it for the temporal DW context. Suppose that a source system contains a measure indicating the interest rate that will be used for the month in a bank. However, the value is given only for the first day of the month and no for the other days. This measure is represented in a temporal DW using the granularity of month. We need to include two kinds of assumptions, point-based and interval-based. The former may be persistence, which indicates that since no modification to data occur, the data remain valid for all granules, i.e., the interest rate is the same for every day of this month. The latter, may refer to upward-heredity, which states that if there exists the specific value for each day of a month, the same value is assigned for a month.

For the forced mapping, it should be noted that coercion functions are always required since a finer time granule can map to more than one coarser time granule, for example, a week to two months. Therefore, measure values to which a finer granule is attached must be distributed. For example, suppose that a salary is paid on weekly basis and this measure is stored into a temporal DW with a granule month. If the week belongs to two months, for example, January and February, a user may specify that the percentage of salary that is assigned for a month is obtained from the percentage of the week contained in the month (for example, 2 days from 7).

4.8.2.3 Temporal types for aggregated measures in temporal DWs

In Section 4.8.1 we analyzed which temporal types can be included in temporal DWs considering those existing in source systems. The discussion in this section will be more limited. Firstly, if source systems are non-temporal, based on explanation given in Section 4.8.1 only data warehouse loading time can be included for aggregate measures without any problems related to granularity. Secondly, if transaction time forms
part of source systems, this time will not be included in a temporal DW when measures are aggregated. The purpose of having transaction time is to analyze changes occurred to individual data, and transaction time for aggregated data will not give useful information for decision-making users.

On the other hand, valid time may exist in source systems for every individual measure. If measure values are aggregated with respect to time, valid timestamps must be adjusted to the corresponding granule in a temporal DW. For example, the valid time of the aggregated measure of salary equal 20 in Figure 4.22 is equal 1 (quarter 1), even though valid time for salary in a source system overlaps also quarter 2. If the aggregated measures do not change during several DW granules, the coalescing operator can be applied. Furthermore, valid time can be used together with data warehouse loading time.

### 4.8.3 Mapping of fact relationships with temporal measures

Depending on analysis needs, temporal measures may represent either events, i.e., something that happens at a particular time point, or states, i.e., something that has extent over time. In the following example, we only refer to aggregated measure values whose valid time is represented as an instant with granularity month. Nevertheless, the results may be straightforwardly generalized if valid time is represented by a period.

A fact relationship in the MultiDimER model corresponds to an n-ary relationship in the ER model. Measures as attributes of this relationship are mapped to the ER model in the same way as temporal attributes of a level. Therefore, each measure is represented as a multivalued composite attribute. Figure 4.23 a) shows the mapping to the ER model of the fact relationship with temporal measures from Figure 4.2.

Mapping this fact relationship to the relational model in first normal form (1FN) gives two tables. In Figure 4.23 b) we only show the table with the Sales measure since the other table with the Quantity measure has similar structure. However, if additional information is available, this model can be simplified. For example, if all measures are calculated with respect to the same valid time, they can be represented in one table and tuple timestamping can be applied. In this way, the two tables may be converted in one having foreign keys, Sales and Quantity attributes, and one attribute for representing valid time.

The object-relational model creates also a separate table based on the following rule:

**Rule 9:** A fact relationship with temporal measures is mapped to the object-relational model by creating a new relation that includes as attributes the references to the surrogate keys of the participating levels. In addition, every measure is mapped into a new temporal attribute according to Rule 5.

An example of the tabular object-relational representation is given in Figure 4.23 c).

However, even though the object-relational model allows to represent the changes in measure values for the same combination of foreign keys, in practice it may be not well suited for aggregations related to time. The objects created for every measure contain two-level nesting: one for representing different measure values for the same combination of foreign keys and another for representing a temporal element. Therefore, it is more difficult to express aggregation statements related to time accessing the
4.8. TEMPORAL MEASURES

Figure 4.23: Temporal measures: a) the ER representation, b) the relational table for the Sales measure, and c) the object-relational representation.

For choosing the object-relational representation, it is important to consider physical-level features of the particular object-relational DBMS. For example, in Oracle 10g if we use nested varying arrays, the timestamps cannot be indexed and comparisons for valid time must be developed by programming. On the other hand, if two nested tables are used, indexing and comparisons are allowed improving query formulation and execution. However, for accessing the measure and its corresponding valid time two nested tables must be joined in addition to a join with the main table containing foreign keys. Therefore, depending on the specific features of the object-relational DBMS, the relational representation may be more adequate in order to represent in a more “balanced” manner all attributes that may be used for aggregation purposes.

Additionally, the inclusion of transaction time and/or data warehouse loading time for non-aggregated measures and data warehouse loading time for aggregated measures requires the corresponding mapping specified in Section 4.5.
4.9 Metamodel of the temporally-extended Multi-DimER model

We give next a metamodel for the temporally-extended MultiDimER model. This model is based on the metamodel presented in Figure 2.4 for conventional DWs.

As shown in Figure 4.24, a dimension is comprised of either a level, or one or more hierarchies. A hierarchy contains several related levels. These levels are associated through child-parent relationship. Levels include attributes, some of which are key attributes used for aggregation purposes.

A temporal level is a level for which the application needs to keep its time-varying characteristics. This is captured by the TemSup attribute in Figure 4.24, which may include different temporal types for a level and/or for attributes. We allow valid time (VT), transaction time (TT), or bitemporal time (BT) coming from source systems (if available) and DWLT generated by DBMS of a temporal DW. Further, valid time of a level is represented by its lifespan (LS).

The relationship between levels may also be temporal independently of whether the levels are temporal or not. This is indicated in Figure 4.24 by the attribute called
TempSup for the Connects relationship between child and parent levels. The model allows to include VT, TT, BT, and/or DWLT for this relationship.

Additionally, the relationship between two levels (the Connects relationship in Figure 4.24) is characterized by cardinalities, which indicate the minimum and the maximum number of members in one level that can be related to a member in another level. The snapshot cardinality is the cardinality in a instant of time whereas the lifespan cardinality represents this cardinality over the members’ lifespan. Both kinds of cardinalities include their minimum and maximum values expressed in the child and in the parent roles.

A dimension is temporal if it has at least one temporal hierarchy. A hierarchy is temporal if it has at least one temporal level or one temporal relationship between levels. Since temporal hierarchies (respectively dimensions) can combine temporal and non-temporal levels (respectively hierarchies), we call a hierarchy (respectively dimension) fully temporal when all its levels and relationships between them (respectively hierarchies) are temporal. It is called partly temporal when it contains at least one non-temporal level or one non-temporal relationship between levels (respectively one non-temporal hierarchy).

A fact relationship represents an n-ary relationship between leaf levels with \( n > 1 \). If it contain attributes called measures, they must be temporal, i.e., they must include a temporal type(s). For non-aggregated measures, TT, VT, and DWLT are supported unlike for aggregated measures where only the latter two can be included.

Table 4.2 gives the summary of temporal types that are allowed in the MultiDimER model.

<table>
<thead>
<tr>
<th>Table 4.2: Temporal types allowed in the MultiDimER model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>lifespan</td>
</tr>
<tr>
<td>Valid time</td>
</tr>
<tr>
<td>Transaction time</td>
</tr>
<tr>
<td>Data warehouse loading time</td>
</tr>
</tbody>
</table>

Notice that the metamodel for the temporally-extended MultiDimER model (Figure 4.24) differs in several elements from the metamodel presented for conventional DWs (Figure 2.4):

- The inclusion of the derived attribute called Temporal for Dimension (respectively for Hierarchy) allows to store the information whether a dimension (respectively a hierarchy) is temporal.

- Levels as well as attributes have an additional attribute (TempSup) representing the type of temporal support if it exists.

- Measures being attributes of fact relationships are always temporal.
• The relationships between child and parent levels forming a hierarchy may have temporal support indicated in the attribute TemSup in the Connects relationship.

• Since the relationship between child and parent levels can be temporal and characterized by snapshot and lifespan cardinalities, the metamodel includes in the Connects relationship corresponding minimum and maximum values for these cardinalities.

4.10 Summary of mapping rules

In this section we present all mapping rules defined throughout the previous sections.

• Rule 1: A temporal type representing an instant is mapped to an attribute of date or timestamp type.

• Rule 2: A temporal type representing a set of instants is mapped to a multiset attribute of date or timestamp type.

• Rule 3: A temporal type representing a period is mapped to an attribute of a structured type composed of two attributes of date or timestamp type.

• Rule 4: A temporal type representing a set of periods is mapped to a multiset attribute of a structured type consisting of two attributes of date or timestamp type.

• Rule 5: An attribute with temporal support is mapped to the object-relational model as a multiset attribute of a structured type composed of two attributes: one for representing the value and another one for the associated temporal type.

• Rule 6: A level is mapped to a relation containing all its attributes and an additional attribute for a key. If a level has LS support, an additional attribute as specified by Rules 3 or 4 should be included.

• Rule 7: A non-temporal many-to-one relationship between child and parent levels is mapped to the object-relational representation by including a parent key in the child level table.

• Rule 8: Firstly, a structured type composed of two attributes is defined: one attribute for representing surrogates of the parent level and another one for the corresponding temporal type. Secondly, this structured type is used for defining a simple or a multiset attribute depending on whether the cardinality between child and parent levels is many-to-one or many-to-many, respectively. Let us call this attribute TemRel. Finally, one of two possible object-relational representations can be used:

1. Creating a new relation that contains an attribute for the surrogate keys of the child level and the TemRel attribute.

2. Extending the relation corresponding to the child level with the TemRel attribute.
4.11. RELATED WORK

- **Rule 9**: A fact relationship with temporal measures is mapped to the object-relational model by creating a new relation that includes as attributes the surrogate keys of the participating levels. In addition, every measure is mapped into a new temporal attribute according to Rule 5.

4.11 Related work

The necessity to manage time-varying data in databases has been acknowledged for several decades, e.g., [49, 94]. However, no such consensus has been reached for representing time-varying multidimensional data. Works related to temporal DWs raise many issues, for example, the inclusion of temporal types in temporal DWs (e.g., [1, 26]), temporal querying of multidimensional data (e.g., [133, 161]), correct aggregation in presence of data and structural changes (e.g., [43, 81, 133]), temporal view materialization from non-temporal sources (e.g., [215]), evolution of a multidimensional structure (e.g., [19, 43, 133]), or implementation considerations for a temporal star schema (e.g., [17]). Nevertheless, very little attention has been drawn to conceptual modeling for temporal DWs and its subsequent logical mapping.

The works reviewed next fall into four categories. First, the works describing different temporal types that may be included in temporal DWs; second, proposals of conceptual models for temporal DWs; third, works referring to different granularities, and finally works related to logical-level representation.

The inclusion of different temporal types in temporal DWs is briefly mentioned in several works. Most of them consider valid time [19, 17, 43, 133, 176, 215]; some of them mention that it is easy to incorporate transaction time [133, 161] or explicitly provide bitemporal time support [127, 104]. However, they usually consider transaction time as time where a fact is current in DW, whereas in our model transaction time as well as valid time are incorporated from source systems. Only Brucker and Tjoa [26] discuss the inclusion of valid time, transaction time, and data warehouse loading time for active data warehouses. However, unlike our approach, they limit the usefulness of these temporal types for only active DWs and do not provide a conceptual model that includes these types.

A more extensive analysis of temporal types for temporal DWs is given by Abelló and Martín [1]. Taking into account different types of sources that integrate data in a temporal DW, they discuss the inclusion of transaction time and valid time in temporal DWs. Valid time is calculated based on transaction time of either a DW or a source system. The exception is made for source systems based on temporal DBs, considering only one temporal type, for example valid time, and converting another one, for example transaction time, into a user-defined attribute. However, we consider that transaction time from source systems can be useful for temporal DW applications. Further, transaction time cannot be used as a possible approximation of valid time, since data can be included and be current in DBs after its validity has expired. Abelló and Martín [1] do not take into account the possible existence of user-defined time attributes in source systems, which may serve for establishing valid time in temporal DWs. Moreover, in addition to transaction time and valid time, we include in the model data warehouse loading time indicating since when the data has been current in
temporal DWs.

Several works are dedicated to conceptual modeling of temporal DWs. Body et al. [19] define a conceptual temporal DW model that allows a member to have several valid member versions for a given time (when valid time overlaps). They include a temporal relationship that establishes an explicit link between two member versions and represents the roll-up function. Since a dimension is a set of member versions and a set of temporal relationships between these members, a temporal dimension is considered as a directed graph where nodes are member versions and arcs are relationships.

Eder et al. [43] propose a temporal multidimensional model called COMET; it allows to represent changes at the schema and instance levels. The model includes valid time for members and for relationships between members forming hierarchies. They include a list of constraints to ensure the integrity of their model. In order to reduce incorrect OLAP results due to the dimension changes, their model includes transformation functions given by the user. The COMET model was extended by Koncilla [104] including transaction time of a temporal DW.

Mendelzon and Vaisman [133] propose a temporal multidimensional model that reuses results from the temporal DB community. They formally define temporal dimension schema and temporal instances as well as a temporal fact table, which are used for defining a temporal multidimensional database. Using valid time they build a TO-LAP query language that allows the user to choose the way data should be aggregated considering its changes in time.

Pedersen et al. [161] extend the basic multidimensional model by temporal support. Their model allows the inclusion of valid time as well as transaction time. These temporal types can be used to express changes in dimension members including their representation, in hierarchy links, and in fact-dimension relationships.

On the other hand, Ravat and Teste [176] define a DW model using an object-oriented approach; it allows to integrate temporal and archive data. Temporal data are used for storing the detailed data evolution while archive data store the summarized data evolutions.

In general, these models formally describe the temporal support for multidimensional models, allowing to express changes in dimension members, hierarchy links, and in fact relationships. However, none of them propose a graphical representation based on a multidimensional view of temporal data that can be used for communication between users and designers. Further, they do not consider different aspects as proposed in this work, for example, a hierarchy that may have temporal and non-temporal levels linked with either temporal or non-temporal relationships, the inclusion of different temporal types for measures, and the problem of different time granularities between source systems and temporal DWs. In addition, these models do not provide an associated logical representation.

There are many works in temporal DBs related to transformations from finer to coarser (or vice versa) granularities, e.g., [16, 37, 212]. However, in this section we only mention some of them\(^\text{13}\). For example, Dyrsen [42] defines mappings between different granularities as explained in Section 4.8.2.1 while Bettini et al. [16] and Merlo et al. [134] refer to the problem of conversion of different time granularities as

\(^{13}\)More detailed references can be found, for example in [16, 37].
4.11. RELATED WORK

well as handling data attached to these granules. Bettini et al. [16] propose calendar operations that allow to capture the relationships existing between time granularities. They define point- and interval-based assumptions that can be used, respectively, for data conversion between the same or different time granularities. Merlo et al. [134] consider the transformation of time-varying data for the ISA relation in a temporal object-oriented data model. To ensure the adequate transformation between a type and subtypes represented with different time granularities, they introduce and classify coercion functions.

Multiple time granularities for measures and dimensions are implicitly considered by Eder et al. [43]. In their approach imprecision may be introduced since the user must give mapping functions for measure distributions between different temporal versions of dimension members.

Even though the aspect of managing data with multiple time granularities is widely investigated in temporal DBs, this is still an open research in temporal DWs. In this work, we consider only the different time granularities between source systems and a temporal DW.

Regarding logical representation for temporal DWs, Bliujute et al. [17] introduce a temporal star schema that differs from the classical one by the fact that the time dimension does not exist; instead the rows in all tables of the schema are timestamped. They compare this model with the classical star schema taking into account database size and performance. They conclude that the temporal star schema facilitates expressing and executing queries, it is smaller in size, and it does not keep redundant information.

Physical storage structures for temporal DWs are proposed by Martín and Abelló [127]. To every timestamped attribute is associated a structure for storing historical data with valid time and transaction time. A separate structure is used for storing current data that include all temporal attributes and the begin timestamps for valid time and transaction time. However, they consider temporal DWs as bitemporal databases requiring all data to be timestamped without giving to users the possibility to choose which temporal changes are important to represent in a temporal DW. Further, their approach create many tables, which may affect system performance.

Given the lack of a satisfactory solution for a logical representation of temporal DWs, we briefly review logical models for temporal DBs with the goal to adapt some of these ideas for the logical representation of temporal DWs.

One approach for logical-level design of temporal DBs is to use normalization. Temporal functional dependencies have been defined, e.g., [93, 209, 210, 211]. New temporal normal forms, e.g., [209, 210], or extensions of conventional ones, e.g., [93], have been proposed. Most of these approaches rely on the first normal form (1NF). However, the non-first normal form (NF2), e.g., [8], was proposed for solving the well-known limitations of the first normal form for modeling complex data. The NF2 allows structured domains, collection domains, and relation-valued domains, and these are also included in the SQL:2003 standard under the name of object-relational model [130, 131]. In addition, leading DBMS vendors (e.g., Oracle, Informix) have also included object-relational features.

Clifford et al. [36] distinguish temporally-grouped and temporally-ungrouped historical data models. The former corresponds to attribute-timestamping models using
complex domains in NF2, the latter to tuple-timestamping models represented in INF. Although these two approaches allow to model the same information, they are not equivalent: while a grouped relation can be ungrouped, for an ungrouped relation there is not a unique grouped relation. Wijsen [211] considers that the approach given in [36] has difficulties in managing time-varying data due to the absence of an explicit group identifier. Further, Clifford et al. and Wijsen [36, 211] consider that temporally-grouped models are more expressive.

Another approach for logical-level design of temporal DBs is based on mapping conceptual models. While this is the usual practice for conventional (i.e., non-temporal) database design, to the best of our knowledge only Gregersen et al. [66] and Snodgrass [190] propose such an approach for obtaining a logical schema from a temporal conceptual model. Gregersen et al. [66] describe a two-steps mapping of temporal ER diagrams to (1) a surrogate-based relational schema and to (2) a lexically-based relational model. Snodgrass [190] proposes two step-mapping: first, non-temporal conceptual schemas are mapped to non-temporal relational tables. Then, he augmented the conceptual schema with the temporal support, which later on is included also in the logical level either as part of the tables or as integrity constraints.

In general, the approach for mapping timestamped elements is to create a table for each entity type that includes lifespan, a separate table for each timestamped monovalued attribute, and one additional table for each multivalued attribute, whether timestamped or not. This approach produces a significant number of tables since entities and their time-varying attributes are represented separately. It is not intuitive for expressing the semantics of the modeled reality.

4.12 Summary

Bringing together two research areas, data warehouses (DWs) and temporal databases, allows to combine the achievements of each of them leading to the emerging field of temporal data warehouses. Nevertheless, neither DWs nor temporal DBs have a well-accepted conceptual model that can be used for capturing users' requirements. To establish a better communication between designers and users, in this chapter we presented temporal extensions of the MultiDimER model. We included temporal support for levels, attributes, relationship between levels forming a hierarchy, and measures giving their conceptual and logical representations.

First, we discussed the inclusion in our model of valid and transaction time coming from source systems and the data warehouse loading time generated by a temporal DW. Next, we referred to the conceptual and logical representations of levels that include temporal attributes and lifespan. We also discussed three different cases for temporal hierarchies: (1) non-temporal relationships between temporal levels, (2) temporal relationships between non-temporal levels, and (3) temporal relationships between temporal levels. When temporal relationships are included, the model allows to represent the snapshot and lifespan cardinalities indicating the number of members of one level that can be related to members of another level in every time instant and over its lifespan, respectively.

For each case, we included the MultiDimER representation and specify constraints.
4.12. SUMMARY

Then, we presented transformation of our model to the ER model. We also specified the mapping rules to the object-relational model based on the SQL:2003 standard. In some cases, we included as an example the Oracle syntax indicating the physical features that should be considered during implementation.

Finally, we refer to temporal measures. We analyzed two different situations when the time granularity for representing measures in temporal DW is either the same or coarser than the one in source systems. For the former, we presented several cases justifying the inclusion of transaction time, valid time, or bitemporal time from source systems and data warehouse loading time generated in a temporal DW.

On the other hand, the different time granularities in source systems and a temporal DW require not only a transformation for time granularities but also an adequate handling of aggregations for measures. We discussed several proposals that already exist for temporal DBs that can also be used for temporal DWs. Further, we referred to different temporal types that may be included for aggregated data, i.e., valid time and data warehouse loading time.

Proposing the inclusion of temporal types in a conceptual model allows to include temporal semantics as an integral part of temporal DWS. In this way, temporal extensions offer more symmetry to multidimensional models allowing to represent changes and the time when they occur for all elements of a DW. Afterwards, logical and physical models can be derived from such a conceptual representation.

The translation of the constructs of the MultiDimER model to the ER models allows better understanding of their semantics. Further, the transformation of the ER model into operational data models is well understood (e.g., [48]) and this translation can be done using usual CASE tools. However, the ER model provides less convenient conceptual representation of time-varying attributes, levels, and relationships than the MultiDimER model. The latter contains less elements, it clearly allows to distinguish which data changes should be kept, and it leaves outside of user’s concerns some more technical aspects.

On the other hand, the proposed rules from our model to the object-relational model allow to assist the implementers who use the MultiDimER model for conceptual design of temporal DWs. Further, these mappings consider the particularities of the different elements of a multidimensional model as well as the specificities of current DBMSs.

Object-relational model allows to represent in a better way temporal levels and hierarchies than the pure relational model. In the former model a level and its corresponding temporal attributes are kept together while the relational model produces a significant number of tables with well-known disadvantages for modeling and implementation. Further, unlike the relational model, for representing a relationship between levels forming a hierarchy the object-relational model gives a designer several alternatives. Thus, he can choose the one considering semantics and physical level features of the particular object-relational DBMS. On the other hand, the relational model is more adequate for representing temporal measures. It considers in the same manner all attributes including the ones that represent time, thus it facilitates aggregation procedures.

The proposed mapping may vary according to the expected usage patterns, for example, data mining algorithms, and specific features of the target implementation.
system. For example, a user may choose a multidimensional tool-specific storage (e.g., Analytic Workspace in Oracle 10g) instead of relying on more general solutions as the ones proposed in this paper.
Chapter 5

Methodology

Data warehouse technology has been widely adopted because many organizations recognize the benefits of using DW data to support decisions. However, there is still lack of a systematic approach for DW design. On the one hand, well-known practitioners informally describe phases that they use for developing DWs [94, 86, 100]. On the other hand, the scientific community proposes a variety of approaches for developing DWs. Nevertheless, they either include features that are particular for the specific conceptual model used by the authors or they are very complex. This situation has occurred since the necessity of building DW systems that reach users’ expectations was ahead of methodological and formal approaches for DW development as the one we had for operational databases [112].

In this chapter we propose a methodological framework for designing conventional, spatial, and temporal DWs. Since DWs are databases devoted to analytical processing [22], our approach is in line with traditional database design.

After considering the current State of the art, we refer to the usual DW lifecycle diagram slightly modifying it by the inclusion of the phase of conceptual modeling. Then, we present a motivating case study that we use throughout this chapter to show the applicability of the proposed methodology and to facilitate the comprehension of different steps.

The subsequent sections include our proposal of three different methods for requirements specifications and conceptual modeling of DWs. Depending on whether users or source system data are the driving force for requirements specifications, we propose, respectively, demand-driven and supply-driven approaches. We also show that the combination of both approaches can provide a realistic analysis scenario that satisfies users’ requirements. The latter approach we called demand/supply driven.

For each approach, we present its general as well as more detailed descriptions. We also illustrate using examples how they can be used in real-world applications. We finish the presentation of the proposed approaches by the inclusion of recommendations in which situations they work better, and indicating their advantages and disadvantages.

Next, we refer to logical and physical design phases. In both we include features related to DW structures as well as to extraction-transformation-loading (ETL) processes. We present an example of transformation of the conceptual multidimensional schema to the logical schema using the SQL:2003 standard. Then, we identify several physical features that can considerably improve performance of analytical queries. We
give examples of their implementation using Oracle 10g.

Finally, considering that spatial as well as temporal DWs play an important role for expanding and improving analysis and for better representing temporal changes to DW data, respectively, we extend the proposed methodology by the inclusion of spatial and temporal data. To achieve this extension, we modify the phases of each method for the requirements specifications and the conceptual modeling and refer briefly to the logical as well as the physical design.

This chapter is organized as follows. Section 5.1 describes methodologies used for design of conventional, spatial, and temporal databases. Section 5.2 presents different approaches that currently exist for the data warehouse design. Section 5.3 refers to the proposed methodology while Section 5.4 includes a motivating case study. The different methods for requirements specifications and conceptual design are given in Section 5.5. Sections 5.6 and 5.7 provide insights about logical and physical design phases for conventional DWs. Section 5.8 presents modifications to the methodology allowing the inclusion in DWs of spatial and temporal data. Finally, Section 5.9 surveys related works and Section 5.10 presents summary of this chapter.

5.1 Database design

5.1.1 Conventional database design

The classical approach for database design consists of two parallel activities: modeling the structure of the database and designing database applications [48]. Even though these two activities can be presented separately, they are closely intertwined. Both activities must be developed during each phase of database design. Next we briefly summarize these phases [48].

- **Requirements gathering and analysis**: this phase allows to obtain the necessary information about the user needs for having the system [181]. A large number of approaches for requirements specifications and analyses has been developed both by academia and practitioners; for example, Winter and Strauch [213] mention that in 1995 not less than 28 techniques and technique combinations for information requirements analysis existed. In general, these techniques help to elicit necessary and desirable system properties from the prospective users and/or project managers, homogenize requirements, and assign priorities to them (i.e., separate necessary from "nice to have" system properties) [213]. During this phase the active participation of users will increase customer satisfaction with the delivered system and avoid corrections of errors, which can be very expensive if the subsequent phases are already developed.

- **Conceptual design**: the main purpose of this phase is to produce the abstract representation of users’ data requirements. In the database environment, the most used conceptual model is the entity-relationship (ER) model proposed by Chen [33]. Alternatively, object-oriented modeling techniques can also be applied based on the UML notation [23]. Conceptual design can be done using two different approaches according to the system complexity and developers’ experience:
5.1 DATABASE DESIGN

- **Top-down schema design**: the requirements of different users are merged before the design process begins and one schema is build. Afterwards, the separation of the views corresponding to the individual user's requirements can be proposed. This approach can be difficult and expensive for large databases and non-experienced developers.

- **Bottom-up schema design**: a separate schema is build for each group of users with different requirements and later, during the view integration phase, these schemas are merged and form a global conceptual schema for the entire database. This is the approach used for large databases.

  - **Logical design**: during this phase the conceptual schema is mapped into the logical schema of the chosen database model, for example relational, object-relational, or object-oriented. To ensure an adequate logical representation suitable mapping rules must be specified. They ensure that the constructs included in the conceptual model can be transformed to the appropriate structures of the logical model.

  - **Physical design**: in this phase database elements are described in terms of database management system particularities including specific storage issues, access paths, indices, among others.

The first two phases, data requirements and conceptual design are the most critical ones since they can considerably affect users' acceptance of the system [102]. Indeed, these two phases determine the adequate relationship between the real world and the software world, i.e., between the users' needs and what the modeled system will offer.

5.1.2 Spatial and temporal database design

Unlike conventional databases, there is not yet a well-established methodology for spatial and temporal database design. In general, the four phases described above, i.e., requirements specification, conceptual, logical, and physical design are considered for both, spatial and temporal, databases (e.g., [90, 186, 190, 214]).

When modeling spatial databases, in many cases conceptual models use additional elements to represent spatial data. Then, the spatially-extended conceptual schema is translated to the logical and physical schemas using corresponding mapping rules. On the other hand, in many situations due to the lack of a well-accepted conceptual model for spatial database design, the phase of the conceptual design is skipped and the logical design is conducted based on users' requirements; the obtained logical schema is later on transformed into the physical schema.

A similar approach can be used for temporal database design [64, 66], when temporal support is included in the conceptual model that later on, is mapped into logical and physical models. Another approach [190] initially ignores all temporal aspects when developing a conceptual schema. After the full design is complete, the conceptual schema is augmented with the temporal elements. Similarly, the logical design is developed in two stages. First, the non-temporal conceptual schema is mapped into the non-temporal relational schema using the already existing mapping strategies, for
example, [48]. In the second stage of logical design, each of the temporal element is included either as part of the tables or as integrity constraints.

5.2 Current approaches for data warehouse design

5.2.1 Data warehouse lifecycle

The development of a DW is a complex and costly task that demands highly-skilled professionals to ensure its successful realization. Since different activities are performed during the project development, their content and their sequence of execution should be clearly specified. However, even though many works exist in the area of project management referring to aspects such as different methodologies [35, 98, 170] or the human behavior [141], very few of them relate to the development of DWs [30, 34, 100, 164]. To our knowledge only Kimball et al. [100] describe in detail the whole lifecycle of DW development. Figure 5.1 presents the different phases they proposed.

![Figure 5.1: DW lifecycle diagram.](image-url)
5.2. CURRENT APPROACHES FOR DATA WAREHOUSE DESIGN

The lifecycle shown in the figure lays out the general phases that are important for the DW project. It indicates the sequence of different tasks without establishing time frames for their realization. The three different paths starting from the business requirement definition and ending with the deployment phase (from left to right) are called technology, data, and application tracks, respectively. Every phase represents a complex activity that may contain different sub-phases. The detailed explanation is given in [100]; in the following we only present a brief description of them.

- **Project planning:** it addresses the definition and the scope of the DW project. It also focuses on resource and staff requirements, project tasks, assignments, duration, and sequencing. The project planning depends on business requirements; this is indicated by the two-way arrow between these two phases.

- **Business requirements definition:** the objective of this phase is to understand and specify business requirements. This phase differs significantly from operational database design since users from different management levels can participate expressing their particular business needs. The requirements specification phase establishes a foundation for all future project activities; it has a major impact on the success of DW projects [185] since it directly affects the technical aspects as well as DW structures and applications.

- **Data track:** it refers to data modeling on different levels.
  - **Dimensional modeling:** the authors propose here a logical-level design. They present a DW schema using a set of star and/or snowflake representations based on the requirements specification and data analysis of relevant operational systems.
  - **Physical design:** in this phase the definitions of the physical structures, indexing and partitioning strategies, among others, are determined. The proposed physical structures highly depend on the specific DBMS used for the DW implementation.
  - **Staging area design and development:** this phase allows to create the extraction, transformation, and loading (ETL) processes and data structures required for their successful execution. They are needed first to extract data from source systems, then to perform all necessary transformations that ensure adequate matching between source and DW data considering data structure and content. Finally, the loading processes are created that allow the initial population as well as the subsequent loads of data from source systems into the DW.

- **Technology track:** this track requires analysis of technical requirements leading to selection and implementation of technological support for the DW.
  - **Technical architecture design:** the overall framework of the current technical environment and the required one for responding to user needs is established. Several aspects must be considered, such as storage capacity, calculation and analytical capabilities, internet access, among others.
— **Product selection and installation**: using the previously-designed architecture, specific components are selected, such as hardware platform, database management system as well as data access and analysis tools. Afterwards, the product installation and testing follow.

- **Application track**: this track is responsible for gathering and implementing user functional requirements.

  - **End-user application specification**: the description of required reports, calculations, and analysis scenarios is performed.
  
  - **End-user application development**: the implementation of end-user applications is realized. Different kinds of applications may be required, for example, static report, scenarios for dynamic data manipulation, among others.

- **Deployment**: this phase ensures convergence of technology, data, and applications. It requires also to develop an education strategy for end-users before giving them access to the system.

- **Maintenance and growth**: in this phase a plan for maintaining and growing the DW is elaborated. The former should ensure that processes and procedures could be correctly executed. The growing plan should consider new users' requirements since the business analysis needs can vary depending on many factors, such as changes in marketing, new competing organizations. In this way, the development lifecycle will be repeated leveraging and building upon the one that has already been established in the DW environment.

- **Project management**: this phase ensures the realization and coordination of DW project phases and activities. Different tasks may form part of project management with the objective of monitoring project status, problem tracking, and control of required changes in project activities.

In many cases the lifecycle proposed by Kimball *et al.* [100] should be customized to address unique needs of an organization. For example, Carneiro and Brayner [30] as well as Pereira and Becker [164] modify and extend this lifecycle allowing the DW project to be developed by the internal staff that has good knowledge about databases but lack the expertise in DW development.

### 5.2.2 Data mart and data warehouse design

A data warehouse includes data of an entire enterprise. This data is used by users belonging to high management levels in order to support the strategic decisions of the organization. However, since the decisions may also be taken in other organizational levels that refer to some specific business areas, for example, sales, financing, only a subset of data contained in a DW is required. This subset is called a **data mart**. It has a similar structure to a DW but it is smaller in size.

Similar to the design of operational databases, there are two major methodologies for the design of a data warehouse and related data marts [100]:
5.2. CURRENT APPROACHES FOR DATA WAREHOUSE DESIGN

- **Top-down design**: the requirements of users on the different organizational levels are merged before the design process begins and one schema for the entire DW is built. Afterwards, separate data marts are tailored according to particular characteristics of each business area or process.

- **Bottom-up design**: a separate schema is built for each data mart taking into account requirements of the decision-making users responsible for the specific business area or process. Later, these schemas are merged forming a global schema for the entire DW.

The planning and implementation of an enterprise-wide DW using the top-down approach is an overwhelming task for most organizations in terms of cost and duration. It is also a challenging activity for designers [100] because of its size and complexity. On the other hand, the reduced size of data marts allows to earn back their building cost in a shorter time period and to facilitate the design and implementation processes.

Leading practitioners usually use the bottom-up approach for developing DWs [84, 100]. However, the latter should be built considering future integration of data marts into a whole DW. Therefore, it is necessary to establish a global DW framework to facilitate the subsequent merging process. The lack of this global framework may cause the DW project to fail [152] since different data marts may contain the same data using different format or structures. Different frameworks can be applied. For example, Imhoff et al. [84] first develop the global DW data model. Then, they build a prototype of a data mart mapping its structure into the DW data model. The mapping process is repeated for each subsequent data mart.

On the other hand, Kimball et al. [100] propose a framework called the **DW bus architecture**, in which each dimension and measure shared between different data marts must be “conformed”. A dimension is “conformed” when it is identical in each data mart that requires its presence. Similarly to conformed dimensions, a measure conforms to the overall enterprise model if the same terminology is used for representing its content across all data marts. By defining conformed dimensions and measure, new data marts may be brought into the DW in an incremental manner ensuring their compatibility with already-existing data marts.

5.2.3 Design phases

Most authors agree that the phases established for developing operational databases, i.e., requirements specification, conceptual, logical, and physical design, can be used for developing data warehouses or data marts (e.g., [34, 61, 82, 114, 169, 183]). Nevertheless, some of the proposals for DW design consider that the development of DW systems is rather different from the development of conventional operational database systems. On the one hand, they include additional phases, such as workload refinement [61] or the extraction-transformation-loading process [114]. On the other hand, they provide different methods for the requirements specification phase, to which we refer in the next section.
5.2.4 Methods for requirements specifications and DW design

In the DW context, elements that are fundamental for multidimensional models, i.e., facts with associated measures, dimensions, and hierarchies should be clearly specified. Most of the existing proposals consider all these elements as a part of requirements specifications. Further, the common practice is to establish order in which they should be included: facts-measures-dimensions-hierarchies [20, 21, 29, 82, 138, 165].

Different approaches can be used for the requirements specification depending on whether users, business goals, or operational source systems are used as a driving force. This leads to the creation of the so-called user-driven, business-driven, or data-driven approaches, respectively. Additionally, a demand/supply-driven approach considers both the business (or user) requirements specifications and the underlying source systems. Next, we give a brief description of these approaches. More detailed descriptions of specific techniques or methods are given in Section 5.9 devoted to the related works.

User-driven approach This approach considers that users play a fundamental role during the requirements analysis and must be actively involved in the elucidation of relevant facts and dimensions [58, 114]. Users from different levels of organization are selected. Then, different techniques, such as interviews or facilitated sessions are used to specify the information requirements [21, 84, 152].

Business-driven approach This approach considers that users often are not able to clearly formulate their particular demands. Therefore, the derivation of DW structures should start from analysis of either business requirements or business processes.

Business requirements specification provides a description of user needs considering business goals, thus starting from the highest level of the organization. Then, the process of subsequent refinements is conducted until identifying the necessary multidimensional elements. Therefore, the obtained specification will include requirements of users at all organizational levels aligned with the previously established business goals.

On the other hand, the analysis of business processes requires to specify different business services or activities that ensure to produce a particular output. Since different elements participate in these activities, they may be considered as dimensions. Further, decision makers need metrics to evaluate business activities, which may be considered as measures in the DW schema.

Other names used for this approach are a process-driven [113], goal-driven [59], or requirements-driven method [169].

Data-driven approach In order to obtain the DW model, the underlying source systems are analyzed. Some of the proposed techniques require conceptual representations of the operational source systems [20, 28, 60, 59, 138] mainly based on the ER model [20, 28, 60, 138]. Other techniques allow to use relational tables for representing source systems. These source schemas should exhibit a good degree of normalization [59] in order to facilitate the extraction of facts, measures, dimensions, and hierarchies. In general, the participation of users is not explicitly required [112], however, in some techniques users should either analyze the obtained schema to confirm the correctness of the derived structures [20] or identify facts and measures as a starting point for the
5.3. A METHODOLOGY FOR CONVENTIONAL DATA WAREHOUSES

design of multidimensional schemas [61, 82, 138]. After schema creation, users can specify their information requirements by selecting items of interest [86, 213].

**Demand/supply-driven approach** This approach is the combination of business- or user-driven and data-driven approaches. Demand indicates business or user data requirements while supply refers to the availability of data in source systems. In the ideal situation these two parts should be equal, i.e., all information that users (business) require for analysis purposes should be supplied by the data included in source systems. This approach is also called top-down/bottom-up analysis [22] or no name is given at all [84, 165].

5.3 A methodology for the design of conventional data warehouses

From the previous section we can notice that the variety of existing methods, specially for the requirements gathering phase, can be confusing for designers even for those with expertise in the area. Some of these methods include very specific steps that only apply for the particular solution proposed by the authors (e.g., [61]). In our approach, we consider a more general methodological framework for the DW development providing designers different options depending on their needs.

We adopt the phases proposed by Kimball *et al.* [100] for the DW development as shown in Figure 5.1. However, in our approach the data track [100] corresponds to the traditional database design phases similar to several proposals [34, 61, 82, 114, 169, 183]. Therefore, we include the phases of conceptual, logical, and physical design as shown in Figure 5.2.

In spite of the similarity between the database and data warehouse design phases, their descriptions should be modified to better represent the specific features of DWs. In the next sections we refer to the DW design methodology considering the phases in Figure 5.2. For the requirements specification, we propose three different approaches. We combine the description of the phases of requirements specification and conceptual design. We consider that presenting them together will help to better understand the examples given for each approach for requirements specification. Further, since Chapter 2 was already dedicated to the aspects of the conceptual design, there is not need to refer to this phase in this section. We only briefly refer to the phase of the logical design since in Chapter 2 we presented this design specifying mappings between the conceptual and the logical schemas. The physical level design depends on the specific features of the DBMS; therefore, we only present some aspects that should be considered in this phase.

![Figure 5.2: Data warehouse design phases.](image-url)
Notice that the phases in Figure 5.2 do not depend on whether the top-down or the bottom-up approach is used, since the difference between both approaches consists in merging and separating schemas, respectively, as explained in the previous section. Whether the top-down or the bottom-up approach is used depends on many factors, such as professional skills of the development team, the size of the DW, users' motivation for having a DW, financial support, among others. For example, if the user motivation is low, the bottom-up approach can deliver a data mart faster and less costly. As a consequence users can dynamically manipulate data using OLAP tools or create new reports; this may lead to increase a user acceptance level and improve motivation for having a DW. From now on, we will use the term data warehouse meaning that the explained concepts apply also to data marts if not stated otherwise.

5.4 A motivating case study

To ensure a better understanding as well as to show the applicability of the proposed methodology and the different approaches used for the requirements specification, we use as an ample the design of a DW for a university focusing mainly on analysis related to research. Since the real scenario contains a considerable amount of detailed information, we use a simplifying one to facilitate the understanding and to focus more on the methodological aspects.

In the last decades universities everywhere have experienced great changes due to different factors, such as decrease in funding, organizational changes, technological breakthroughs, globalization, and increased competition. In order to cope with these issues, universities need easy-to-use information technology solutions that provide accurate analysis enabling more strategic decision making.

Most universities passed through changes in the models of information systems. First, they had one centralized system, which was used for operational purposes as well as for delivering all necessary reports required to support management process. These systems grew significantly in size and complexity due to the increasing interest to have more data that facilitates the execution of different services, such as students' admission, inscriptions to courses, administration tasks. However, the management of a huge amount of data led to well-known problems. In addition to that, different universities' units required more specific data according to services they provided. This situation drove towards decentralization. Nevertheless, the decentralization resulted in numerous uncoordinated and unintegrated information systems, which could not provide a global view of data for decision makers.

In recent years, it has become very important for all levels of the organization to access, analyze, and share information, i.e., to have a global vision of the information. Therefore, there is a clear tendency to create an information-rich decision-making environment that could reach users on different management levels. As a consequence, currently, many universities have initiated projects for implementing DW systems.

The main areas of activities at universities are higher education and research. They include different services that are provided to students (e.g., courses), to society (e.g., applied research), to science in general (e.g., fundamental research), etc. Analyzing data related to these services may help to discover a wider range of alternatives for
increasing revenue, reducing costs, and improving the overall situation. For example, the analysis of data related to academic progress could reduce the cost of instruction and the time to graduation by more accurately identifying academic difficulties. Similar analysis of data related to research activities could help to understand directions of changes required for improving and/or expanding research and obtaining more funding.

Universities are usually divided into faculties representing general fields of knowledge, for example, medicine, engineering, science. These faculties comprise different departments allowing to focus on more specialized sub-fields; for example, the faculty of engineering may have departments of civil engineering, mechanical engineering, materials engineering, computer science, among others. This organizational structure allows to better support educational activities. It also provides support for research giving the university professors the possibility to create research teams within a faculty or a department and offering the necessary logistics for the management of research projects.

In many universities research activities have been intensified and also expanded to cover multi-disciplinary (cross-faculty) areas. To support growing and multi-disciplinary tendencies in research, new structures, such as research centers or research institutes, have been created. They are the main force in developing research activities. They are highly autonomous and may make decisions about the selection and way of conducting the research projects. University professors from different faculties or departments as well as other external personnel can be members of the research centers or institutes.

In our example, a new university entity has been created, i.e., a research department that serves as a bridge between high-level executives (e.g., rector, the research council) and research developers, and between the research developers and external community. For example, the research department is responsible for the evaluation of research activities, for the elaboration of strategic research plans, for promoting research activities and services, among others.

To ensure high-quality research, universities define strategic plans for research. They may refer to different aspects, such as to establishing strategic research areas, to promoting research on international forums, to integrating research and education, or to contributing in the economic and social development through activities of technology transfer and services for the society, among others.

The establishment of strategic areas allows to reflect core strength and ambitions indicating long-term potential and relevance. These areas are the focus of institutional initiatives and investments. Each area may be represented by one faculty (or department) or several faculties (or departments) when referring to a multi-disciplinary area. Based on the institutional research strategy, faculties as well as research centers and institutes identify their own research priorities.

In our example, we will refer to one of the university goals of increasing the participation of the researchers in the international forum within the previously-defined strategic research areas. The development of a strategy for achieving this goal requires information about current situation related to, for example, international publications, participation in international conferences or in international projects. International publications include conference proceedings, journals, or books. Participations in international conferences consider costs and roles of the participants, for example, invited speakers, authors. The project is considered international when the university
professors participate in projects managed in other countries or professors from other countries are invited to form a part of the research team in the university projects.

5.5 Requirements specification and conceptual design

The requirements specification phase is the step before the conceptual design. Being one of the earliest steps of system development and thus entailing significant problems if faulty or incomplete, requirements analysis should attract particular attention and should be comprehensively supported by effective methods [213]. However, on the one hand, not much attention has been paid to the requirements analysis phase in DW development [129]. On the other hand, the variety of existing approaches for requirements specification leads to the situation that many DW projects skip the requirements analysis phase; instead, they concentrate on technically-oriented issues, such as database modeling or query performance [153]. As a consequence, it is estimated that more than 80% of DW projects fail to meet user needs [185] and do not deliver the expected support for the decision-making process.

Requirements specification determines, among others, what data should be available and how it is organized as well as what queries are of user interest. The requirements specification phase should lead to discovering the essential elements of the multidimensional model, i.e., facts with associated measures, dimensions, and hierarchies [20, 21, 29, 82, 138, 165], which are required to facilitate future data manipulations and calculations.

In this section we present our proposal for the phase of data requirements specification that leads to creation of the multidimensional conceptual schema. For the latter, we use the MultiDimER model described in Section 2.2, which uses the notations presented in Figure 2.2. For the requirements specification we propose different options that consider as a driving force business or user demands, existing data in the underlying operational systems, or both business demands and data. For each of the approaches, we first present its general description, then we refer in more detail to different phases and include examples to illustrate the process. We also give the recommendations indicating when it is more appropriate to use the specific approach and we discuss its advantages and disadvantages.

5.5.1 Demand-driven approach

5.5.1.1 General description

In the demand-driven approach, the driving force for developing a conceptual model are business or user requirements. These requirements express the organization goals and needs that a DW is expected to address to support the decision-making process. Since users on different management levels (for example executives, managers, professionals) can require DW support, the identification of key users is an important aspect. Stakeholders, users, business domain experts, and also an enterprise plan will help a developer team to understand the purpose of having a DW and to determine specific
5.5. REQUIREMENTS SPECIFICATION AND CONCEPTUAL DESIGN

analysis needs. The gathered information serves as basis in the elaboration of the initial DW schema. However, in order to support organizational decisions, different elements present in this schema should correspond to real data. Therefore, the verification of whether the data required by users is available in source systems is necessary before logical and physical schemas will be developed. During the process of verification, mapping and general description of required transformations between elements of the DW schema and data in source systems are realized. In the case of lacking some data items, a modification of schema considering the users’ viewpoints should be made. Modifications to the schema may lead to changes in the mappings.

5.5.1.2 Phases

![Diagram of Phases]

Figure 5.3: General phases in the demand-driven approach for a) requirements specification and b) conceptual design for conventional DWs.

The general phases of the demand-driven approach for gathering requirements and creating a conceptual model are shown in Figure 5.3 a) and b), respectively. We present the sequence of phases without indicating different iterations that may occur. For example, the elaboration of the initial schema may require to go back to the previous step in order to refine the analysis needs. Since we provide a general framework, we consider that the proposed solution can be later on detailed and tailored to the particularities of a specific DW project. We give next a general description of each phase.
Requirements specification As shown in Figure 5.3 a) the requirements specification includes two phases of identifying users and determining analysis needs. The latter represents a complex task that includes several steps.

- **Identify users:** since a DW is intended to provide an enterprise-wide decision support infrastructure, users on different hierarchy levels of the organization should be considered [113]. *Executive users* on the top organizational level may express their business needs indicating which information is required, most probably in summarized form. They help in understanding high-level objectives and goals, and overall business vision [100]. *Management-level users* may require more detailed information and they can refer to a more specific area of the organization. They can provide more insights into the business processes or the tactics used for achieving business goals. Finally, *professional users* can be responsible for the specific section or services and they may demand more generalized as well as specific information related to their area of interest. Further, users representing different entities from horizontal (for example, departmental) division should also be considered. This will help in understanding the overall view of the project and its scope. Therefore, even though, the DW project may start focusing on a specific business area or process for the initial development, the identification of potential users should consider vertical (hierarchical) as well as horizontal (departmental) divisions [100].

- **Determine analysis needs:** analysis needs help to understand what data should be available to respond to users’ expectations for having a DW system. This phase comprises several steps:
  
  - **Define, refine, and prioritize goals:** the starting point to determine analysis needs is the consideration of business goals. Successful DW projects assume that the company goal is the same for everyone and the entire company will therefore be pursuing the same direction. Therefore, clear specification of goals is essential to guide user needs and convert them into data elements [112] or to find critical business processes required for goal accomplishments. Since users on different management levels participate in requirements specification, analysis needs may be expressed considering the general goals as well as more specific ones. The latter should be aligned with general goals to ensure a common direction of the overall enterprise development. The goal gathering process can be conducted as interviews, facilitating sessions, or brainstorming based on different approaches, such as those specified in [22, 59, 100, 152]. The list of goals should be refined, for example, some goals may be considered as sub-goals, other goals should be combined because of their similarity, or discarded because of their inconsistency. For every goal subsequent phases as showed in Figure 5.3 a) are realized. We propose two different approaches based either on more specific definition of user demands (upper part in Figure 5.3 a)) or on modeling of business processes (lower part in Figure 5.3 a)).
  
  - **Detail user demands:** additional interviews with the specific users focusing on more precise goal definitions will be conducted. The performer of the
interviewing process should be able to pose the questions tailored to uncover data warehouse issues in more detail. Other techniques than interviews can also be used, such as workshops, questionnaires, or prototyping.

- **Model business processes**: the accomplishment of the goals is closely related to business processes. Therefore, these business processes should be determined for every specific goal. Since a business process is a group of services or activities that together create a result of value for a customer [77], in the next step the identification of these services or activities takes place. Some activities or services may be complex and may require subsequent divisions in smaller tasks. Activities or services include data required for their realization. This data should be considered in the requirements gathering phase since it may form part of the future multidimensional model.

Business processes can be considered implicitly and informally as described in [100] or a more formal business process model can be created as proposed in [21].

- **Establish facts with measures and dimensions**: the previous phase of more detailed specification of user demands or modeling business processes should finish when the specification of facts (focus of analysis) with measures and dimensions (perspectives of analysis).

**Conceptual design** The creation of a conceptual schema consists in three phases as shown in Figure 5.3 b).

- **Elaborate initial schema**: well-specified business or user requirements lead to the clearly-distinguishable multidimensional elements, i.e., to facts, measures, dimensions, and hierarchies. Therefore, the first approximation of a conceptual schema can be developed. We recommend to use a conceptual model to improve the communication with non-expert users in order to determine whether the proposed solution responds to users' analysis needs. This conceptual schema represents only stakeholder and users' information requirements without considering data availability in source systems.

- **Determine data availability and specify mappings**: data in source systems determines whether the proposed conceptual schema can be transformed into logical and physical schemas and be fed with the data required for analysis. All elements included in the conceptual schema are checked against the data items in source systems. This process can be very time consuming if the underlying source systems are not documented, denormalized, or represent legacy systems. The result of this phase is the specification of mappings for all elements of a multidimensional schema that match with data in source systems. This specification includes also the description of required transformations, if they are necessary.

- **Elaborate final schema and refine mapping**: if data is available in source systems for all elements of the conceptual schema, the initial schema is considered as a final schema. However, if some elements cannot be included in the multidimensional schema, a new schema should be elaborated and be presented
to the users for their acceptance. The changes to the schema may require the corrections of existing mappings, for example, deletions of some of them.

5.5.1.3 Example

In this section we show an example of applying the demand-driven approach for the specification of requirements and creation of a conceptual model for a university DW.

Identify users In the first step we identify users on different management levels that make strategic, tactic, or operational decisions related to research process. Three group of users were established:

1. Executive: rector, his advisors, and the research council responsible for establishing general politics for the research.

2. Management: representatives of the research department including the sections responsible for support of research activities and administrative tasks, of promotions, and of evaluation and strategic analysis.

3. Professional: representatives of different research entities including research centers and university departments.

Determine analysis needs The next step of determining analysis needs starts with the specification of the goals. Finding goals is a complex and creative task [21] and may require many sessions with users. General goals usually expressed by executive users should be refined. Management and professional users help considerably in this refinement process. For example, the general goal of increasing the international participation in strategic research areas expressed by executive users may be refined to a more specific goal or sub-goal; management-level users consider that the promotion of these research areas will help to increase the number of international projects; professional-level users include more specific goals of increasing the number of international publications and participation in international conferences as well as in international projects. Notice that both kinds of users express the sub-goal of increasing number of international projects. Therefore, these two sub-goals are merged into one.

In the next step either more detailed information needs from users are specified or a business process model is created. In this way we can identify which data is required for analysis to support the decision process leading to the accomplishment of the proposed goal. Since we focus on one simple goal of increasing the international participation of researchers, several activities are considered as already stated by professional-level users. These activities are international publications and participation in international conferences as well as in international projects. Notice that we have obtained this information by interviewing users. We did not create a university process model since the two main processes, i.e., education and research, are very complex and may require sophisticated techniques to ensure their correct representation, such as described in [21] for the business process model of higher education process.
For the analysis of international publications different users express their particular data needs. For example, the users from research units require detailed specifications of publications including the author, title, discipline to which it belongs, and publication type, i.e., books, conference proceedings, or journals. They also consider the analysis by month and by year as well as by academic semester and academic year. On the other hand, the users from the research department require additionally data to analyze publications by faculties to which belong researchers, by different disciplines, and publishers. The publishers are important in order to consider the subscription of the university library to the electronic editions of articles, journals, or books of the publisher. The executive-level users, i.e., rector and the research council, require information about the total number of publications by year to analyze whether the number of international publications is increasing or decreasing.

The specification of requirements for international conferences and international projects was also elaborated during the interviews with users. For the international conference the users would like to analyze data related to the cost associated to participations of researchers in conferences, for example, the cost of registration to conferences, transportation, lodging, and other expenses. The analysis should be also available on more general level of departments or faculties. Similar to the previous case, users require two different calendars, i.e., common and academic. Further, additional information about different kinds of participation is required, i.e., invited speaker, organizer, author, or member of the audience. This could help to evaluate whether university strategic research areas are acknowledged in the international forum, for example by inviting speakers to conferences.

Finally, for international projects the users need to focus on the salary earned by each researcher and number of hours that each researcher is involved in the specific project. Additional analysis of the salary and hours by different disciplines as well as by research centers or departments is also important. For this analysis users want to include only a common calendar.

Elaborate initial schema Since users consider three different focuses of analysis related to international publications, international conferences, and international projects, three multidimensional schemas are elaborated.

The multidimensional schema for international publications is shown in Figure 5.4. It includes several important features. Firstly, since all kinds of publications including publications in the national forum are important for analyzing researchers activities, we did not create another dimension for the international publications; instead we include an additional hierarchy level called Type, which allows to select whether international or national publications are analyzed. Secondly, users should be able to analyze publications according to the discipline to which they belong. We include this information as a separate level to indicate its importance for analysis purposes. Thirdly, a publication belongs to one of the three different categories, i.e., journal, book, or conference proceedings. Therefore, we use a generalized hierarchy to model it. Fourthly, since academic semesters can start and finish in the middle of the month,
two time hierarchies should be created. Finally, researchers are affiliated to one or more departments, each of which belongs to one faculty.

Another multidimensional schema for international conferences is shown in Figure 5.5. According to users' requirements, we have established different measures, i.e., registration, transportation, lodging, and other expenses. Similar to the previous model for the international publications (Figure 5.4), we have included the Type level to indicate whether the conference is international. We have also referenced both Time dimensions, i.e., representing common and academic calendars. We have created an
additional dimension (the Role dimension in Figure 5.5) that includes the information about different kinds of participation, i.e., invited speaker, organizer, author, or member of the audience.

Notice that even though both figures, i.e., Figures 5.4 and 5.5 contain the same elements (e.g., Researcher, Month, and Type), in the global schema (Figure 5.7) they will be represented only once.

Finally, the multidimensional schema for analysis of international projects was elaborated as shown in Figure 5.6. Since projects can belong either to research centers or to departments, a generalized hierarchy is used to model it. Similar to publications (Figure 5.4) we use the Type and Discipline levels to indicate whether the project is international and to which research area it belongs, respectively.

The three multidimensional schemas (Figures 5.4, 5.5, and 5.6) presented above can be merged to form a data mart that contains data to support decisions related to increasing the participation of the researchers on the international forums. The resulting schema in shown in Figure 5.7.

Since in real-world applications the data mart or the data warehouse schema may contain much more elements, it may get difficult to visualize all levels and fact relationships together. In order to know which levels are common for different fact relationships, we propose to use a matrix as shown in Table 5.1. The columns of the matrix contain the names of the fact relationships and the rows include the names of levels comprising hierarchies or one-level dimension. Every cell of the matrix containing the symbol ✓ indicates that the level is related (directly as a leaf level or indirectly forming a hierarchy) to the fact relationship. This matrix is similar to the one proposed
Figure 5.6: Initial multidimensional schema for the analysis of international projects.

by Kimball et al. [100] for conformed dimensions. However, we extend the concept of conformed dimensions to conformed levels, since in our model levels can be reused between different hierarchies.

Table 5.1: Specification of conformed levels.

<table>
<thead>
<tr>
<th>Schema</th>
<th>Publication facts</th>
<th>Conferences facts</th>
<th>Projects facts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Researcher</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Department</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Faculty</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Publication</td>
<td>✓</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Journal</td>
<td>✓</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Discipline</td>
<td>✓</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>Type</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Figure 5.7: An example of a data mart schema.
CHAPTER 5. METHODOLOGY

Check data availability and specify mappings  The next step in the proposed methodology is to check data availability in source systems for all elements included in the DW schema. We do not present here the structures of the underlying source systems due to their complexity. In the best scenario, the source systems should be documented using a conceptual model, such as the ER model. Thus, the mapping between two conceptual schemas could be realized. However, in the absence of the conceptual representation of source systems, their logical structures can be used instead. In our example, the revision process revealed several aspects that require additional analysis:

- For the multidimensional schemas for analysis of publications and conferences given in Figures 5.4 and 5.5, we do not have the information of time represented by academic cycles (semester, scholar year). However, we do have the exact date of the publication and of the conference. Therefore, during the ETL process this information can be included to specify the required time frame.

- The Type level, which indicates whether the publication, conference, or project is international or national must also be derived from the operational databases. This information is included as a flag in the corresponding tables of operational databases.

- The data required for the Discipline level exists in the operational database for the description of the projects, however, it is not included for publications. The latter contain keywords that can be used for deriving the discipline. However, users should make decision whether this derivation can be done automatically or manually since the automatic derivation may introduce misleading information in the case when the same keyword belongs to several disciplines.

- In the operational databases the information about journals as well as conference proceedings does not include publishers. Therefore, the decision must be made whether this level will be included only for the book (it is present in the operational databases) or additional effort should be made to find this information and to include it in the data mart.

Elaborate final schema and refine mappings  After revision and additional consulting with the users about required changes due to the content of data sources, the final schema and specification of corresponding mappings were elaborated.

5.5.1.4 Recommendations

The demand-driven approach requires an intensive participation of users from different organizational levels. In particular, the support of executive-level users is important to define business goals and needs. The identification of key users for requirements specifications is a very important and difficult task. It requires to consider several aspects:

- Targeted users should be aware of the overall business goals and objectives to avoid the situation where requirements represent the personal perceptions according to the user profession or to the identification with a specific business unit [213].
5.5. REQUIREMENTS SPECIFICATION AND CONCEPTUAL DESIGN

- Users who dominate the requirements specification process should be avoided or tempered in order to ensure that information needs of different users can be considered.

- Users should be available and agree to participate during the whole process of requirements gathering and conceptual modeling.

- Users should have an idea what a DW and an OLAP systems can offer. If this is not the case, users should be adequately instructed by giving explanations, showing demonstrations, or using prototypes.

The development team requires high-qualified professionals. For example, a project manager should have very strong moderation and leadership skills [112]. A good knowledge of information gathering techniques or modeling business processes is also required. A DW designer should be able to communicate and to understand non-expert users in order to obtain the required information and later on, present and describe the proposed multidimensional schema. This helps to avoid the situations when users describe the requirements for the DW system using business terminology and the DW team delivers the system specification to the users from a more technical viewpoint difficult to be understood by the users [25].

Before making the decision of which method of the requirements specification is more convenient for the particular DW project, it is important to consider several advantages of the demand-driven approach:

- DW requirements are derived from the business perspective, therefore they provide a comprehensive and a precise specification of the needs of stakeholders from their business viewpoint [113].

- An effective participation of users ensures a better understanding of facts, dimensions, and the relationships existing between them [58].

- Using formal techniques for defining the business process model provides a formal description of the informational requirements of the users [21].

- This approach may allow to raise the acceptance of the system if constant interactions with potential users and decision makers are realized [112].

- External data may be specified for the inclusion in a DW [112].

- Long-term strategic goals can be specified [112].

However, some disadvantages of using this approach can play an important role in determining its usability for a specific DW project:

- The specification of business goals can be a difficult process and its result depends on the applied techniques and skills of the developer team.

- The specification of users' requirements not aligned with business goals may produce a very complex schema that does not support decision processes of the users of all organizational levels.
• Requirements specification based on business processes can get more complicated if these processes cross organization boundaries [113].

• The duration of project tends to be longer than using the supply-driven method [112]. Therefore, the project cost can be higher.

• Informational requirements of users could not be satisfied with the actual information offerings of the source systems [21].

5.5.2 Supply-driven approach

5.5.2.1 General description

The supply-driven approach relies on the data in source systems. It aims at identifying all candidate multidimensional schemas that can be realistically implemented on the top of the available operational databases. These databases are analyzed in an exhaustive manner in order to discover the elements that can represent facts with associated measures. Usually the facts correspond to events that happen at a particular point in time and contain measurements that may be summarized. These events are related to other data that are candidates for dimensions and hierarchies. The identification of facts, measures, dimensions, and hierarchies leads to the initial versions of multidimensional schemas that may correspond to different analysis purposes, i.e., they may form an initial DW schema. Notice that in this stage it is straightforward to obtain the specification of mappings since the resulting multidimensional schemas are based on source data. However, not all facts will be of interest of the decision support, thus user input is required in identifying which facts are important. Users can also refine the existing hierarchies since some of them may be “hidden” in a table. As a consequence the initial DW schema is modified until it becomes the final version accepted by users. The modification of the schema leads to changes in the specification of mappings.

5.5.2.2 Phases

Figure 5.8: General phases in the supply-driven approach for a) requirements specification and b) conceptual modeling, respectively. As for the demand-
driven approach above, we do not show the different iterations that may be required before developing the final DW schema. Each of the phases in Figure 5.8 represents a complex task. In the following, we only present a brief description of them.

**Requirements specification** As shown in Figure 5.8 a), during the process of requirements gathering the phases of identifying source systems and applying derivation process are realized.

- **Identify source systems**: the aim of this phase is to determine which existing operational systems can serve as a data provider for a DW. The external sources are not considered in this stage. An important aspect is to count on system documentation preferably represented using the ER model or relational tables. If several operational systems exist, those considering data quality and stability of their schemas should be selected [62]. It is also important to determine which data can be usefully integrated in order to obtain a complete view of the database domain [62].

- **Apply derivation process**: different techniques can be used for deriving multidimensional elements from operational databases [20, 22, 29, 60, 82, 138]. All these techniques require that operational databases are represented using the ER model or relational tables.

In general, in the first step the facts with associated measures are determined. This can be done analyzing the existing documentation [20, 29, 138] or the database structures [60]. Facts and measures are elements that correspond to events occurring dynamically in the organization, i.e., that are frequently updated. If operational databases are relational, they may correspond to tables and attributes, respectively. If the operational databases are represented using the ER model, facts may be entity or relationship types while measures are attributes of these elements. An alternative option may be the inclusion of users that understand the operational systems and can help to determine which data can be considered as measures [82].

Different procedures can be applied for deriving dimensions and hierarchies. They can be automatic [22, 60], semi-automatic [20], or manual [29, 82, 138]. The former two require knowledge of the specific conceptual models that are used for the initial schema and its subsequent transformations. The process of discovering a dimension or a leaf level of a hierarchy usually starts from identifying the static (not frequently updatable) elements that are related to the facts. Then, the search for other hierarchy levels is conducted. For this, starting with a leaf level of a hierarchy, every one-to-many relationship is revised.

Unlike automatic or semi-automatic procedures, manual procedures allow to find hierarchies embedded within the same entity or table, for example, to find the city and the province attributes in a store entity, or a customer occupation attribute that can be used for grouping purposes. However, either the presence of the system expert who understands the data in operational databases is required or the designer must have good knowledge about the business domain and the underlying systems.
Conceptual design  The specification of requirements allows to proceed with the conceptual modeling that can be divided in different phases as shown in Figure 5.8 b).

- **Elaborate initial schema and mappings:** since facts with measures, dimensions, and hierarchies are already specified during the previous phase, the elaboration of an initial DW schema is straightforward. Similarly to the demand-driven approach, we recommend to use a conceptual model, such as the one proposed in this thesis, in order to facilitate the future communication with business users. In this phase the specification of mappings between source systems and the proposed schema should be also elaborated. This is an easy task since the multidimensional model originates from source systems.

- **Determine user interest:** until now the participation of the users was minimal responding only to the specific designer inquiries. In this phase users are incorporated in a more active role. They usually belong to professional or administrative levels since this kind of users possesses enough knowledge of the underlying systems to understand the proposed schema. The schema is examined in all its details in order to determine what kind of analysis can be done. However, the initial schema may contain more elements than those required for the analysis purposes of decision-making users. Therefore, the exclusion of some elements may be required.

- **Elaborate final schema and refine mappings:** users’ recommendations about changes are incorporated into the initial schema. This final conceptual schema can be transformed later on to logical and physical schemas. The modifications in the schema are also reflected in mappings, if necessary.

### 5.5.2.3 Example

To demonstrate how the supply-driven approach works in real-world situations let us consider the ER schema containing information about professors and their teaching as well as research responsibilities (Figure 5.9). Notice that to simplify explanation we chose a small part of the operational database, therefore, for the example we skip the phase of identifying the source systems.

Apply derivation process  We choose a manual derivation process to provide a more general solution. However, the previously mentioned automatic or semi-automatic methods can also be applied.

Analyzing the schema in Figure 5.9 we can distinguish four many-to-many relationship types with attributes that represent numeric data: the relationship types Teaches, Participates, Does research in, and Works in. They can be considered as facts and associated measures.

The Teaches relationship type includes an attribute indicating week hours (TWeek hours in the figure) that could be used as a measure. This relationship type is associated to two entity types, i.e., to Professor and to Course, which might be considered as dimensions in order to provide different analysis perspectives. The Teaches relationship type also contains dates referring to the academic year (Sem. and Year in the figure).
5.5. REQUIREMENTS SPECIFICATION AND CONCEPTUAL DESIGN

This data may be included in the corresponding Time dimension. Similar analysis was done for the Participates, Does research in, and Works in relationship types.

Other relationship types in Figure 5.9, i.e., Registered and Included in, do not have associated numeric attributes and are characterized by the one-to-many cardinalities. Therefore, they can be considered as links between hierarchy levels where each level is represented by the corresponding entity type in the figure.

An additional analysis was needed in order to consider whether zero-to-many relationship types (Belongs1 and Belongs2) could be handled as a hierarchy.

Elaborate initial schema and mapping The previous analysis allows to build a set of the multidimensional schemas forming a DW as shown in Figure 5.10. Notice that each multidimensional schema can be represented separately, i.e., including only one fact relationship and its corresponding dimensions with hierarchies.

The schema in Figure 5.10 includes four fact relationships. They can be used for different analysis purposes. For example, the Education fact relationship allows to analyze different education activities of professors while the Work fact relationship can be useful to evaluate overall involvement of professors in their corresponding departments.
Figure 5.10: The derived DW schema from the ER schema of Figure 5.9.
For each fact relationship corresponding dimensions (for example, the Professor or the Project dimensions) were associated. A hierarchy Register formed by the levels Course, Department, and Faculty was created as a part of the model following the relationship types with one-to-many cardinalities between corresponding entity types in the ER model.

On the other hand, the Organizational structure represents a generalized hierarchy allowing to analyze projects that either belong to research centers or departments. It was created considering the Belongs1 and Belongs2 relationship types in Figure 5.9. Notice that based on the dates included in the operational data model, we have created two Time dimensions representing common (Start time and End time) and academic calendars (Academic time).

Since the operational schema is very simple, the mapping between two schemas in Figure 5.9 and Figure 5.10 is straightforward: the fact relationships correspond to many-to-many relationship types in the ER model; all one-level dimensions (e.g., Professor) or leaf levels of hierarchies (e.g., Course) are mapped to entity types in the ER model that participate in these relationship types. Each subsequent hierarchy level is mapped to the entity type in the ER schema that is related by one-to-many cardinality. Different dates representing common and academic time must be derived from dates represented as attributes of the relations (Teaches, Participates, Does research in, and Works in Figure 5.9).

**Determine user interest** The initial DW schema as presented in Figure 5.10 is delivered to the users, i.e., to the representatives of different research units. Since they are not interested in academic issues and administrative assignments of professors to departments or to research centers, the Education, Work, and Research center personnel fact relationships were eliminated from the schema. The Time dimension representing academic time was also excluded since it is not required for the remaining fact relationship.

**Elaborate final schema and refine mapping** The schema that includes only the Project participation fact relationship and associated dimensions with hierarchies is finally delivered to the users. The specification includes also the modified mappings that indicate only those elements that are present in the multidimensional schema.

**5.5.2.4 Recommendations**

In the supply-driven approach the participation of the users is not explicitly required [112]. They are involved sporadically either to confirm the correctness of the derived structures or to identify facts and measures as a starting point for creating multidimensional models. Usually, users of the professional or the administrative organizational levels participate since data is represented at a very low level of detail.

On the other hand, this approach presumes the presence of highly-skilled and experienced designers. Besides the usual modeler abilities, they additionally should possess sufficient business knowledge in order to be able to understand the business context and its needs relying mainly on operational data. They should also have the capacity to understand the structure of the underlying operational databases. In many cases they
may be forced to develop a conceptual model to better understand the relationships existing between the different data items.

The supply-driven methodology has several advantages:

- It ensures that the DW reflect the underlying relationships in data [138].
- For the reason that a complete data set is supplied, the DW contain all necessary data since the beginning [165].
- Developing DWs based on already existing operational data model simplifies the extraction and transformation processes [138].
- An enterprise data model may provide a more stable basis for design than user requirements, which may be subject to changes [138].
- Minimal user time is required to start the project [165].
- The development process can be fast and straightforward if well-structured and normalized operational systems exist [112].
- In some cases automatic or semi-automatic techniques can be applied, such as those proposed in [20, 60].

However, the following disadvantages are important to consider before choosing this approach:

- Since the usual practice is not to gather business or users requirements before the DW design process begins, only those business needs which are reflected in the underlying source data models can be captured.
- Company goals and users' requirements are not reflected at all. Therefore, the system either may not answer users' expectations or may handle many unneeded information structures [213].
- The resulting schema has a very low level of granularity since operational data sources are used [112].
- This approach cannot be applied when the logical schemas of the underlying operational systems are huge and hardly understandable or the data sources reside on legacy systems whose inspection and/or normalization is not recommendable [59].
- External data sources are not considered.
- Since it relies on existing data, it cannot be used when long-term strategic goals must be analyzed or specified [112].
- Creating a multidimensional schema based on existing data may not allow to include different kinds of hierarchies existing in real-world situations, since they may be "hidden" as different structures, for example, as ISA relations.
It is difficult to motivate end-users to participate in the process since they are not used to work with large data models developed for and by specialists [213].

The derivation process can be confusing without knowing user needs since some data can be considered as measures as well as dimension attributes. Further, in non-conventional applications, such as that for our university example, facts and measures can be very difficult to distinguish since they may not represent typical updating behavior as in operational databases of other kinds of businesses.

5.5.3 Demand/supply-driven approach

5.5.3.1 General description

The demand/supply-driven approach combines both previously-described approaches that may be used in parallel to achieve optimal design. Therefore, two chains of activities can be distinguished: one that corresponds to business demands and another one that represents the phases for creating schema from operational databases. The schema obtained from the demand-driven approach identifies the structure of the DW as it emerges from business requirements. The schema from the supply-driven approach returns a DW schema that can be extracted from the existing operational databases.

After the initial schemas are elaborated using both approaches, the comparison between them is realized. If the result of matching process between two schemas is satisfactory, the final schema is delivered. This schema will be mainly based on the schema obtained from the supply-driven approach. However, it may contain some modifications in order to better represent users’ needs, for example, separate some data in a level.

Nevertheless, the matching process can reveal that either business demands exceeds the data availability or operational databases provide more analysis scenarios that users did not consider before. In both situations, some actions must be taken to determine the direction of changes in one of the schemas.

5.5.3.2 Phases

The general phases of the demand/supply-driven approach for requirements specification and conceptual modeling are shown in Figure 5.11. In addition to the three initial phases described for the demand-driven and the supply-driven approaches, a new phase of matching process is introduced. This phase compares (or integrates) the initial schemas from the demand and the supply chains.

The comparison or integration process is not an easy task. Different aspects should be considered, such as used terminology, degree of similarity between the two solutions for each multidimensional element, for example, between dimensions, between dimension attributes, or between hierarchy levels. One of the possible solutions is provided by Bonifati et al. [22]. They derive a DW schema using the so-called GQM (Goal/Question/Metric) paradigm [9] for the demand chain. The schema in the supply chain is obtained by transforming the operational ER schemas into a connectivity graph and applying algorithmic exploration of this graph. Finally, the matching process is performed based on the specific metrics that allow to compare and rank
schemas. There are other techniques, for example Giorgini et al. [59] uses Tropos [31], an agent-oriented software development methodology adapting it to the DW particularities. However, both techniques require a highly-technical knowledge about specific procedures proposed by the authors.

During the matching phase user demands may be covered by data in operational systems and there may be no other data to expand the analysis spectrum, i.e., both schemas cover the same aspects of analysis. This is the ideal situation that leads to accept the schema created by the supply chain as a final schema. Some modifications to this schema may be required to represent users' needs with more clarity. The changes to the schema may lead to the changes in mappings and in the descriptions of required transformations.

Nevertheless, in real-world applications it is difficult to find that both schemas will cover the same analysis aspects. Indeed, two other situations may occur:

1. Users demand less information than what operational databases can provide: in this case it is necessary to determine whether users may consider new analysis aspects or eliminate from the schema those fact relationships that are not of user interest. Therefore, another iteration in the demand and the supply chains is required. In this iteration either new users will be involved who could be interested in the new solutions provided by source systems or a new initial schema will be
elaborated eliminating from the analysis some fact relationships and associated dimensions.

2. Users demand more information than what operational databases can provide: similarly to the previous case two situations may be analyzed. On the one hand, the users may reconsider their demands and limit them to those proposed by the supply-chain solution. On the other hand, users may require the inclusion of external sources or legacy systems that were not considered in the previous iteration but contain the necessary data. Therefore, new iterations in the demand as well as in the supply chain may be needed.

The phases of demand and supply chains can be developed in a parallel or in a sequential manner depending on users availability and the development team capacity.

5.5.3.3 Example

Let us suppose that using the demand approach we create the schema for analysis of international projects as shown in Figure 5.6. On the other hand, the supply-driven approach gives the schema shown in Figure 5.10. During the matching process we can clearly see that the latter schema includes additional information not required by the users, i.e., they demand less information than operational databases can provide. Therefore, during another iteration within the demand chain the decision should be made whether analysis related to Education, Research center personnel, and Work fact relationships in Figure 5.10 are important in accomplishing business goals. For example, decision makers from university departments could be interested in having the Education fact relationship in pursuing another university goal of more effective distribution of human resources leading to decreasing course costs.

Notice other similarities and differences between two schemas. In Figure 5.10 the Type and Discipline levels are not presented as hierarchies since they are difficult to distinguish when only analyzing data. The Time dimension in Figure 5.6 indicates the monthly salary and number of hours of the researcher participating in the specific project. On the other hand, the Time dimension in Figure 5.10 contains two hierarchies: Start and End time and measures indicating the number of week hours and salary paid for these hours.

After additional iterations, a final schema is elaborated. It includes the data delivered by the supply chain with the necessary refinements to better represent users' needs (e.g., separating discipline as a level). New mappings are also elaborated as well as a description of required transformations as the ones mentioned above.

5.5.3.4 Recommendations

Since this approach combines the demand-driven and the supply-driven methods, the recommendations regarding users and the development team given previously should also be considered here.

The demand/supply-driven approach has several important advantages. It generates a feasible solution (i.e., supported by the existing data sources) that best reflects the users' goals [22]. It can also indicate missing data in operational databases that
are required to support the decision-making process. If the source systems offer more information than the business users initially demand, this approach helps to expand analysis to the aspects not yet considered by users.

However, the development process is more complicated since both schemas, modeled from definitions of business requirements and derived from the underlying source systems, are required. Additionally, the integration process to "measure" whether data supply covers data demands may require complex techniques [22, 59].

5.6 Logical design

Two aspects are important to consider during the logical design phase: firstly, the transformation of the conceptual multidimensional schemas into the logical representation, secondly, more detailed specifications of the ETL processes considering mappings and transformations indicated in the previous phase. We will refer next to these two aspects.

5.6.1 Logical representation of DW schemas

As explained in Section 2.4.1, the logical representation of a DW conceptual schema is often based on the relational data model [62, 86, 101, 128] using star and snowflake schemas. Many DW applications also include some pre-computed summary tables with aggregated data storing them as so-called materialized views. However, we do not consider such tables to be part of the core logical schema.

Relational representation of a DW schema can also be used in ROLAP systems. Recall that this kind of systems, similar to MOLAP systems, facilitates analytical processes providing flexible interactions with end-users and dynamic manipulations of data contained in DWs. The difference between ROLAP and MOLAP systems consists that the former store data in relational tables while the latter use vendor-specific data structures. Notice that even though MOLAP logical models could be considered as another possible mapping, unfortunately, there is no agreed-upon multidimensional model neither in the scientific community [89, 206] nor in commercial products [135, 83, 147]. Additionally, in Section 2.3 we already discussed in more detail our rationale for using relational representation.

Since in our methodology the conceptual design is considered as a first and an abstract representation of the DW structure, mapping rules between conceptual and logical models are required to obtain a logical schema. It should be clear that for the same logical representation, for example, relational, different mapping rules are used for every specific conceptual model [29, 61]. In Section 2.3 we described general mapping rules from our MultiDimER conceptual model into the relational model. We extended some of them in order to better represent the semantics of different kinds of hierarchies. In this section we apply these rules to obtain relational schemas of the multidimensional conceptual schemas elaborated in the previous phase. In order to provide a more general solution, we use the SQL:2003 standard [130, 131] for defining relational tables.

We will discuss view materialization in the phase of physical-level design.
5.6. LOGICAL DESIGN

We present next an example of mapping into relational tables for the schema in Figure 5.5. First, considering users' analysis needs, query performance, and data reuse, we must decide whether a star or a snowflake representation should be chosen\(^3\). For example, for performance reasons we present the Calendar and Academic time hierarchies in one table (i.e., we de-normalized them) instead of mapping every level to a separate table:

```sql
create table CalendarTime (
    CalTimeSysId integer primary key,
    MonthNo integer,
    MonthName character varying(15),
    QuarterNo integer,
    YearNo integer);
```

```sql
create table AcademicTime (
    AcaTimeSysId integer primary key,
    SemesterNo integer,
    YearNo integer);
```

The Affiliation hierarchy in Figure 5.5 contains three levels: Researcher, Department, and Faculty. Since the Department level is used in another multidimensional schema (Figure 5.6), in order to reuse existing data we decided to use the snowflake representation for this hierarchy, i.e., represent each level in a separate table. We first declare the Faculty level, which then is referenced in the Department level:

```sql
create table Faculty (
    FacultySysId integer primary key,
    FacultyId integer unique,
    FacultyName character varying(35),
    Dean character varying(35));
```

```sql
create table Department (
    DepartmentSysId integer primary key,
    DepartmentId integer unique,
    DepartmentName character varying(35),
    Chairman character varying(35),
    FacultyFK integer,
    constraint facultyFKconstr foreign key (FacultyFK) references Faculty (FacultySysId));
```

The Affiliation hierarchy is a non-strict hierarchy\(^4\), i.e., it contains a many-to-many cardinality between the Researcher and the Department levels. In order to represent this cardinality we use a bridge table as explained in Section 2.4.2. For that, we create a Researcher table and an additional table called ResearcherDepartment, which references both Researcher and Department tables:

```sql
create table Researcher (
    ResearcherSysId integer primary key,
    ResearcherId integer unique,

\(^3\)In Section 2.4.1 we already specified the advantages and disadvantages of using star (de-normalized) or snowflake (normalized) schema for representing dimensions with hierarchies.

\(^4\)In Section 2.4.2 we gave the definition of this kind of hierarchy.
ResearcherName character varying(35),
ResearcherTitle character varying(35));
create table ResearcherDepartment ( ResearcherFK integer,
DepartmentFK integer,
constraint researcherFKconstr foreign key (ResearcherFK) references Researcher (ResearcherSysld),
constraint departmentFKconstr foreign key (DepartmentFK) references Department (DepartmentSysld),
constraint resDepFKconstr primary key (ResearcherFK,DepartmentFK));
In a similar way we define tables for other levels, i.e., for Type, Conference, and Role:
create table Type ( TypeSysld integer primary key,
TypeName character varying(35) unique,
TypeDescription character varying(128));
create table Conference ( ConferenceSysld integer primary key,
ConferenceId integer unique,
ConferenceName character varying(35),
Description character varying(128),
TypeFK integer,
constraint typeFKconstr foreign key (TypeFK) references Type (TypeSysld));
create table Role ( RoleSysld integer primary key,
RoleName character varying(50),
Description character varying(128));
Finally, the fact relationship table is created. It contains all measures included in the conceptual schema and references all participating dimensions. Notice the inclusion of referential integrity constraints in order to ensure the correct data insertion:
create table ConferenceFacts ( ResearcherFK integer,
DepartmentFK integer,
ConferenceFK integer,
CalTimeFK integer,
AcaTimeFK integer,
RoleFK integer,
RegistrationCost decimal(5,2),
TravelingCost decimal(5,2),
LodgingCost decimal(5,2),
OtherExpenses decimal(5,2),
constraint researFKconstr foreign key (ResearcherFK) references Researcher (ResearcherSysld),
constraint depFKconstr foreign key (DepartmentFK) references Department (DepartmentSysld),
 constraint resDepPKconstr unique key (ResearcherFK,DepartmentFK,ConferenceFK,CaTimeFK,AcaTimeFK,RoleFK,RegistrationCost,TravelingCost,LodgingCost,OtherExpenses));
5.6. LOGICAL DESIGN

Department (DepartmentSysld),
constraint confFKconst foreign key (ConferenceFK) references Conference (ConferenceSysld),
constraint calTimeFKconst foreign key (CalTimeFK) references CalendarTime (CalTimeSysld),
constraint acaTimeFKconst foreign key (AcaTimeFK) references AcademicTime (AcaTimeSysld),
constraint roleFKconst foreign key (RoleFK) references Role (RoleTimeSysld),
constraint factPK primary key (ResearcherFK.DepartmentFK,ConferenceFK, CalTimeFK,AcaTimeFK,RoleFK));

The previously-defined tables use primary keys (for example, FacultySysld in the Faculty table or DepartmentSysld in the Department table) whose values must be supplied by users during the insertion process. However, in Section 2.3 we specified the advantages of having surrogates, i.e., system-generated keys. The latter can be obtained by using the facilities of object-relational databases and by declaring typed tables as explained in Section 4.6.2. Recall that typed tables are tables that require some structured types for their definition. They contain an additional self-referencing column keeping the value that uniquely identifies each row. For example, we can define a structured type for representing the Faculty level as follows:

create type FacultyType as (
    FacultyId integer,
    FacultyName character varying(35),
    Dean character varying(35) ref is System generated;
)

Then, we create a typed table Faculty of this structured type:

create table Faculty of FacultyType (
    constraint facultyIdUnique unique(FacultyId),
    ref is FacultySysld System generated);

The clause ref is FacultySysld system generated indicates that FacultySysld is a surrogate attribute automatically generated by the system. The Department table can be declared in a similar way. It will additionally include the reference to the surrogate of the Faculty table:

create type DepartmentType as (
    DepartmentId integer,
    DepartmentName character varying(35),
    Chairman character varying(35),
    ColIRef REF(FacultyType) scope Faculty
    references are checked) ref is system generated;

create table Department of DepartmentType (
    constraint departIdUnique unique(DepartmentId),
    ref is DepartmentSysld System generated);

In this way the Department table contains references to the corresponding rows of the Faculty table. This may help to avoid costly join operations between hierarchy levels.
However, whether the object-relational or the relational model is used depends on the application needs as already mentioned in Sections 3.2.1. While choosing a model, it is also important to consider the physical-level design and specific features of the DBMS as we will explain in Section 5.7.

5.6.2 Defining ETL processes

During the previous phase of the conceptual design we have additionally created a schematic representation of the mappings between the source and target data. We have also specified some transformations that may be necessary in order to match user requirements with data available in the source systems. However, before implementing the ETL processes, several additional tasks must be specified in more detail.

In the logical design phase, all transformations of source data should be considered. Some of them can be straightforward, for example, the separation of the department address into its components (e.g., street, city, ZIP code, etc.) or the extraction of date components (e.g., month, year). Others may require additional decisions, for example, whether to recalculate measure values in the ConferenceFacts table or value of salary in the ProjectFacts table to express them in euros or to use the original currency and to include the exchange rate. It should be clear that according to different real situations, more complex data transformations may be required. Further, since the same data can be included in different source systems, the decision of whether to discard some of the repeated items or to merge them should also be made.

Moreover, users must specify how the changes in source data that correspond to dimension data in a DW will be managed, i.e., whether old and new data will be kept or only the last value is important. This will help the designers to include the necessary structures that allow to represent these changes, if necessary.

Another aspect to consider is refreshing frequencies and methods. Since the DW should represent current data in order to better support the decision processes, not only an initial loading of data from source systems into the DW is necessary, but also subsequent DW data refreshments must be considered. For example, data in a fact table may be required on a daily or a monthly basis, or after some specific event (e.g., after finishing the specific project) while changes in dimension data may be included after their commitment in the source systems. Therefore, users should specify the time frame or the event as a time point to begin the data refreshment.

A preliminary sequence of execution of ETL processes should also be determined. This is required to ensure that all data will be transformed and included checking their consistency. For example, in Figure 5.5 first, the Faculty, then, the Department, and finally, the Researcher levels should be included. Obviously, this order is required to conform to the referential integrity constraints. Similarly, for the fact table, first dimensions data must be populated before that data for the fact table will be loaded.

It should be obvious that ETL processes can be the most difficult part of developing a DW. They can be very costly and time demanding. To facilitate the design and subsequent executions of ETL processes, different commercial tools can be used, such as Microsoft Data Transformation Services, SAS administrator, or Oracle Data Warehouse.

---

5Refer to the previous example in the demand-driven approach.
5.7. PHYSICAL DESIGN

Builder.

5.7 Physical design

Similar to the previous logical level design, we should consider two aspects in the physical design phase: one related to implementation of the DW schema and another one that considers ETL processes. For each of them we only refer to some general physical design issues since others depend on the particularities of the DW as well as specific features of the target database management system (DBMS).

5.7.1 DW schema implementation

During the physical design phase, the implementer converts the logical model into the tool-dependent physical database structure. Physical design decisions consider the proposed logical schema as well as analytical queries or applications specified during the process of requirement gathering. A well-developed physical design should help to improve access to DW data, query performance, DW maintenance, data loading processes, among others. Therefore, a DBMS used for building a DW should include features that assist the implementers in different tasks, such as in managing huge amount of DW data, in refreshing the DW with new data from source systems, in performing complex operations that may include joins with many tables, in aggregating many data items. This assistance can be given by providing different kinds of storage, variety of indices, table partitioning facilities, parallel query execution, aggregation functions, view materialization, among others.

Next we will refer to some of these features showing their importance in the DW context. As an example of a DBMS we have chosen Oracle 10g [148].

5.7.1.1 Storage method

The usual practice in developing a DW is to use relational databases due to advantages already specified in Section 2.3. Relational tables as a basic unit of data storage are mostly organized in a star or a snowflake schemas. Then, DW data can be queried using the classical SQL functions and operators, for example, sum, count, group by, or more sophisticated ones specially defined for DWs, for example, cube or rollup. If more advanced query and calculation capabilities are required than pure DBMSs could provide, specialized systems can also be used, such as ROLAP or MOLAP.

Currently, there is a clear tendency in leading DBMS companies, such as Oracle or Microsoft, to offer an integrated environment that allows to manage data represented in relational tables and in a tool-specific multidimensional format. For example, Oracle 10g integrates the multidimensional storage within relational databases through the so-called analytic workspaces. The data is stored as a LOB (large object) table and can be manipulated by the OLAP engine. Within a single database, many analytic workspaces can be created. These workspaces require the definition of a logical multidimensional schema that later on is mapped to the physical data model and populated with data.

6This kind of tables allows to store a huge amount of unstructured data.
Users may choose whether to perform analysis with data that is entirely stored in analytic workspaces or is distributed between analytic workspaces and relational tables.

Further, Oracle 10g introduces a special object called dimension. It is not mandatory to define it, however, if the application uses a multidimensional schema (for example, a star schema), it can give several benefits in query rewriting, materialized view management, and index creation, among others. The dimension is an abstract concept that requires the existence of the physical dimension table containing the corresponding data. For example, for the Diffusion hierarchy in Figure 5.5, we first create tables for the Type and the Conference levels as follows:

```sql
create table Type(
    TypeSysId number(6) primary key,
    TypeName varchar2(35) unique,
    TypeDescription varchar2(128));

create table Conference(
    ConferenceSysId number(15) primary key,
    ConferenceId number(10) unique,
    ConferenceName varchar2(35),
    Description varchar2(128),
    TypeFK number(6),
    constraint typeFKconstr foreign key (TypeFK) references Type (TypeSysId));
```

Then, we can define the Conference dimension as follows:

```sql
create dimension ConferenceDim(
    level ConferenceLev is (Conference.ConferenceSysId)
    level TypeLev is (Type.TypeSysId)
    hierarchy Diffusion(
        ConferenceLev child of TypeLev
        join key (Conference.TypeSysId) references Type);
```

The ConferenceDim includes two levels, i.e., ConferenceLev and TypeLev indicating to which relational tables and attributes they correspond, i.e., respectively, to Conference.ConferenceSysId and Type.TypeSysId. The definition of hierarchy, i.e., Diffusion, specifies the child-parent relationship existing between levels. Since we use normalized tables, i.e., separate tables for Conference and Type levels, the above declaration includes an additional statement referring to attributes used for the join operation.

However, Oracle establishes a set of constraints that must be fulfilled to ensure that correct results can be returned from queries that use these dimensions. These constraints may be sometimes too restrictive, for example, they do not allow to define non-strict and generalized hierarchies.

In spite of some limitations that can be found in commercial systems, we can say that having an integrated architecture for representing multidimensional and relational data facilitates data administration and management. It provides more capabilities for performing different kinds of queries without the necessity of using separate software and hardware platforms.

---

^7We will refer to these concepts in subsequent sections.
5.7. PHYSICAL DESIGN

5.7.1.2 Fragmentation

Fragmentation, partitioning, or clustering in databases means to divide a table in smaller data sets to better support management of very large data volumes. There are two ways of fragmentation: vertical and horizontal [151].

Vertical fragmentation allows the designer to group attributes of each relation into smaller records; for example, a dimension may be fragmented to have the name and city attributes in one partition and the remaining attributes in another partition. Therefore, if the query requests names, more records can be retrieved into main memory since they are smaller in size after partitioning, i.e., they contain less attributes.

On the other hand, the horizontal fragmentation divides a table in several smaller tables with the same structure, i.e., attributes, but with less records. For example, if certain users' queries require the most recent data and others access the older data, a fact table can be horizontally partitioned according to some time frame (e.g., year) [163].

Therefore, since smaller data sets are physically allocated in several partitions, they considerably facilitate administrative tasks (a new partition can be added or deleted), give advantages in query performance applying parallel processing (if software and hardware support it), or allow to access a smaller subset of data (if the user's selection do not refer to all partitions).

Fragmentation techniques should be chosen during physical DW design. However, to be able to use them, implementers must have a good knowledge not only of the meaning and consequence of having partitioned dimension and fact tables, but also which specific method of partitioning may work better. For example, Oracle [148] provides four types of horizontal partitioning methods, such as range, hash, list, and composite. Each of them has different advantages and design considerations. Oracle gives a detailed description and empirical recommendations in which situations a specific method would work better.

An example of defining a partitioned table, using Oracle, is given next. Based on the supposition that the university is involved in many projects, we create a table for the Project level in Figure 5.10 with two partitions considering whether or not the project is international:

```sql
create table Project (
    ProjectSysId number(15) primary key,
    ProjectId number(10) unique,
    ProjectName varchar2(35),
    Responsible varchar2(64),
    International char(1),
    ...
) partition by range International
    (partition IntProj values ('Y'),
    partition NatProj values ('N'));
```

As we can see using partitioned tables instead of non-partitioned ones addresses the key problem of supporting very large data volumes by allowing the implementer to decompose them into smaller and more manageable physical pieces.
5.7.1.3 Indexing

The most relevant kind of query submitted to DW systems is the so-called star query [97]. Star queries include the references to dimension and fact tables. They impose restrictions on the dimension values that are used for selecting specific measures; these measures are further grouped and aggregated according to the user demands. The major bottleneck in evaluating such queries is the join of the central (and usually very large) fact table with the surrounding dimension tables. Therefore, to deal with this problem in addition to the classical B-tree index, a new kind of indices, the so-called bitmap indices, are used [32, 97, 144, 148].

A bitmap index can be seen as a matrix where each entry corresponds to an address (i.e., rowid) of a possible record and each column to different key values. If the bit is set, it means that the record contains the key value. The advantages of using the bitmap index are greatest for columns, in which the ratio of the number of distinct values to the number of rows in the table is small [148]. For example, a gender column with only two distinct values (male or female) is optimal for a bitmap index and can be defined in Oracle as follows:

```sql
create bitmap index researcherBitmapldx(
on Researcher(Gender));
```

Bitmap indices allow to speed up the access to the fact table performing the so-called star join [97, 148]; it filters out dimension records that satisfy some conditions before accessing the dimension table itself. If the where clause contains multiple conditions, the filtering process is repeated for each condition. Then, bitmaps corresponding to the different dimension values represented in each condition can be combined using AND or OR logical operators depending on the selection condition. The resulting bitmap can be used to extract tuples from the fact table [97].

Therefore, using different kinds of indices according to particularities of DW queries could help to improve system performance.

5.7.1.4 Materialization of aggregates

Another way of accelerating query performance in DWs is precalculating expensive join and aggregation operations and storing the obtained results in a table in the database [97]. This issue has been studied extensively mainly as materialized views and their maintenance problem (e.g., [68, 191]). Therefore, since materialized views represent query results that have been stored in advance, long-running calculations are not necessary when users actually execute SQL statements; instead, the access to the table representing this materialized view is realized.

For example, in the schema created for the analysis of conferences (Figure 5.5), we can create a materialized view that allows to store the aggregations of conference cost according to the conference type and academic year:

```sql
create materialized view ConferenceCostMV (build immediate
   refresh complete
   enable query rewrite
as select T.TypeName, A.YearNo,
```

sum(CF.RegistrationCost), sum(CF.TravelingCost),
sum(CF.LodgingCost), sum(CF.OtherExpenses),
count(*) as cnt, count(CF.RegistrationCost) as cntRegCost
count(CF.TravelingCost) as cntTravCost, count(CF.LodgingCost) as cntLodgCost,
count(CF.OtherExpenses) as cntOthExp
from Conférence C, Type T, AcademicTime A, ConferenceFact CF
where C.TypeldFK = T.TypeSysld and
CF.ConferenceldFK = C.ConferenceSysld and
CF.AcaTimeldFK = A.AcaTimeSysld
group by T.TypeName, A.YearNo;

This example creates a materialized view ConferenceCostMV that computes the total number\(^8\) and value of conference cost (considering four measures included in the schema, i.e., Registration, Lodging, Traveling costs, and Other expenses) presenting them for each conference type and academic year. For this example, we suppose that the table representing the Academic time hierarchy is stored as denormalized structure, i.e., all levels are included in the same table called AcademicTime.

Notice that Oracle requires to define when materialized views should be populated, i.e., when the data will be loaded to the table created by the materialized view statement. In the example the chosen method is immediate (build immediate). The refreshment method must be also specified, i.e., when the data in the materialized view is updated. Oracle provides different options, the one chosen in the example (refresh complete) indicates that after the inclusion of new data in the DW, the whole materialized view is recalculated.

To ensure that a materialized view will be accessed instead of executing the query that created it, the query rewrite mechanism should be enabled (enable query rewrite). Afterwards, Oracle's optimizer is able to automatically recognize when an existing materialized view can and should be used to satisfy a user request. It transparently rewrites the request using the materialized view instead of going to the underlying detail tables.

Nevertheless, the design and management of materialized views is not an easy task. Not all possible aggregations can be precalculated and materialized since this could lead to the phenomena called "data explosion" where the number of aggregates grows exponentially with the number of dimensions and hierarchies [188, 194]. In the last years much research has focused on the problem of defining which aggregates should be pre-calculated in order to find the trade-off between storage and performance (e.g., [57, 191, 177]). However, the proposed algorithms are usually highly technical and in many cases very sophisticated making their application by DW implementers difficult.

The selection of the loading and refreshment methods for materialized views require a good knowledge not only about the DW project requirements but also about solutions provided by the particular DBMS. In addition, the query rewrite mechanism may impose many restrictions on the issued queries, which when not fulfilled, make the rewriting process to fail, i.e., the query is executed and calculated against the detail data instead of using the materialized view.

In general we can say that even though management of materialized view is not

\(^8\)This data is required by Oracle syntax.
CHAPTER 5. METHODOLOGY

an easy task, using them may improve the query performance. Selecting an adequate DBMS can facilitate the processes of creating, populating, refreshing, and using materialized view. However, an additional overhead related to storage capacity and management of additional structures should be considered.

5.7.2 ETL processes and staging area implementation

In the physical level design all the required extraction, transformation, and loading processes must be defined in a more detailed, ready-to-execute manner. Notice that this decomposition into three different processes is done only logically since the same software component can be in charge of all steps [89].

The extraction process must be implemented not only for the first loading of data into a DW but also for the subsequent refreshments. For the latter full or incremental extraction methods can be used. Using the former method, the data is extracted completely from the source systems and included in the DW. Depending on the particularities of the DBMS, different options exist, for example, export mechanism, ODBC or OLE DB, among others.

On the other hand, in the incremental extraction method, only the data that has changed after an event or a time point (already specified by the user during the logical phase design) is extracted. This kind of extraction requires a mechanism to identify which data was modified. Several methods exist; in Section 4.3 we describe different kinds of source systems and briefly refer to methods of extracting only data that has changed. For example, the entire table can be copied from the source system into an intermediate storage and compared with the data from the previous extraction using the set difference operator usually provided by SQL. Another method can include additional structures, such as timestamps in source systems for indicating the last data change. Some DBMSs can provide facilities for the change detection. For example, Oracle 10g includes the so-called change data capture mechanism that allows to identify and capture data that has been added to, updated in, or remove from Oracle relational source systems; these changes are stored in a relational table and are available for use by applications or users.

Extracted data may require some transformations. The transformation process may be a simple SQL instruction or may require complex programming effort. All transformations specified during the conceptual and logical design phases should be now implemented.

Additional storage may be needed for the transformation process in order to check or change the incoming data into the required output. This storage is the commonly-called data staging area [101]. For example, the inclusion in the staging area of lookup tables for the Conference table might help to determine whether the conference name from source systems is valid.

Therefore, considering different extraction methods and transformation needs, the decision should be made whether the staging area is required. In the affirmative case, this additional storage should be designed and created.

Finally, the process of loading data into the DW may be implemented using avail-

---

9Figure 5.5 shows a conceptual model, which was translated to the logical model in Section 5.6.1.
A METHODOLOGY FOR SPATIAL AND TEMPORAL DWS

able in the specific DBMS mechanism, such as Oracle’s SQL*Loader or export/import facilities.

All implemented processes must be scheduled and processed in a specific order. Depending on the success or failure of the process or part of it, the result must be tracked and subsequent, alternative processes might be started. Therefore, it is important to include adequate exception handling in each component process as well as in the whole ETL process. This will help to avoid unpleasant situations of not giving the required data to the users.

Implementation of the extraction–transformation–loading (ETL) process may be a very complex, time-consuming, and costly task. To succeed, it is important to have well-documented specification of mappings and transformations obtained from the previous phases of conceptual and logical design, a good knowledge of the DBMS used for implementing a DW, an ability and/or knowledge to understand the physical structures of the underlying source systems, and high-quality programming skills.

5.8 A methodology for design of spatial and temporal DWs

The proposed methodological framework for the conventional DW design is general enough so it can also be applied for the spatial and the temporal DW design as we will see in the next sections. However, the previously-presented descriptions of all phases explicitly refer to traditional data types without considering spatial and temporal support. In this section we refer to these phases extending them by the inclusion of spatial and temporal data. We also propose some modifications that allow to better guide the designers and the implementers in developing spatial and temporal DWs.

5.8.1 Spatial extension

5.8.1.1 Spatial support for elements of multidimensional models

The spatially-extended MultiDimER model allows to include spatial data for levels, levels’ attributes, and measures as explained in Chapter 3. Additionally, different topological relationships can form part of a model if two or more spatial dimensions are present. Therefore, according to analysis needs different elements can be considered for the inclusion of spatial data.

5.8.1.2 Requirements specification and conceptual design

The proposed design methodology for conventional DWs provides three different approaches for the requirements specification that lead to the creation of conceptual schemas. Since these approaches rely on users’ (business) analysis needs, source data, or both, we should consider both, whether the users are able to express their requirements regarding spatial support and whether spatial data is included in the source systems.

See Section 3.6 for more details.
For example, if the source systems include spatial data, e.g., spatial databases or geographic information systems, users may be already familiar with concepts related to spatial data including their manipulation and some kinds of simple analysis. Therefore, this kind of users may be able to express which spatial data they require in order to exploit features of multidimensional models and to perform different kinds of spatial analysis, such as described in Chapter 3.

On the other hand, source systems may not include any spatial data. In this case, users may not be familiar with aspects related to spatial data features and manipulation. Therefore, they may require a simple spatial support that allows to visualize data in a space leading to discovering patterns that otherwise are difficult to find. For example, users may require to empower the analysis process visualizing living places of company’s customers.

**Demand-driven approach** Two different possibilities exist for the requirements specification and conceptual modeling of spatial DWs using the demand-driven approach (Figure 5.12). The upper line in the figure refers to the situation when the users are familiar with concepts related to spatial data. These phases are the same as the ones for the conventional DW design (Figure 5.3). However, since we suppose that from the beginning of the requirements gathering process users are able to express the specific analysis needs referring them to spatial data, the elaborated initial schema already includes spatial elements. The following phase of checking data availability may include an additional activity of getting spatial data from external sources if the required spatial support do not exist in source systems.

![Figure 5.12: General phases in the demand-driven approach for spatial DWs.](image)

In the case when users are not familiar with spatial data management, we propose the inclusion of an additional phase as can be seen in the lower part in Figure 5.12. The first three phases are developed as for the conventional DW. During the next phase, designers present to the users the initial schema and request them for indications of the required spatial support. First, one-level dimensions should be considered choosing whether a level, its attributes, or both should be represented spatially. Then, every hierarchy level is analyzed in a similar way. If there are more than two spatial dimensions, designers can help the users to determine whether some topological
relationships between dimensions may be of users' interest. In the affirmative case, a specific topological relationship should be included in the fact relationship to indicate a predicate for the spatial join operations\textsuperscript{11}. Finally, the inclusion of spatial measures may be considered. Similar to the previous case, the phase of checking data availability may require the access to external sources since spatial data may not be present in the underlying source systems.

Notice that the phases indicated by the lower line in Figure 5.12 can also be used when users have knowledge about spatial data, but they (or designers) prefer first to express their needs related to non-spatial elements and afterwards, include spatial support.

**Supply-driven approach** Similar to the previous approach, we refer to two cases. Nevertheless, since operational databases are the driving force for this approach, now we consider whether these databases are spatial.

If source systems include spatial data, the same phases as for the conventional DW design can be applied (the upper line in Figure 5.13). However, a special derivation process should be designed to create an initial schema with spatial elements. We believe that currently this derivation process should be conducted manually and must rely on the designer knowledge of business domain and spatial DW concepts. This is since to our knowledge semi-automatic or automatic procedures for spatial DWs as the ones developed for conventional DWs do not yet exist.

![Figure 5.13: General phases in the supply-driven approach for spatial DWs.](image)

However, if spatial data is not included in the source systems, the first four phases as specified for the conventional DWs (Figure 5.8) are conducted. After determining users' interest related to the conventional DW schema, a new phase of adding the spatial support is realized (the lower line in Figure 5.13). Notice that this support is only considered for the previously-chosen elements of the multidimensional schema. The analysis which elements should be spatially represented can be conducted in a similar way as explained for the demand-driven approach above. Since spatial support does not form a part of the underlying operational systems, external sources should

\textsuperscript{11}For more detailed explanation refer to Section 3.6.
deliver required spatial data. The corresponding mapping should be included as a part of the final schema.

Notice that the phases indicated by the lower line in Figure 5.13 can also be used when the source systems include spatial data but the derivation process is complex. In this case, spatial elements present in a multidimensional schema will be mapped to spatial data of source systems.

**Demand/supply-driven approach** Figure 5.14 includes modified phases for this approach. If source systems include spatial data and users have knowledge about it, the first three phases are realized as explained above for the demand-driven and the supply-driven approaches. Then, the matching process will consider spatial data that is included in both schemas, i.e., obtained from the demand and the supply chains. If the result of this matching process is satisfactory, the final schema is delivered (the upper line in Figure 5.14). In other case, additional iterations may be necessary as indicated for the conventional DW design.

On the other hand, if source systems do not include spatial data or users are not familiar with the concepts related to spatial data, all steps until the matching process of schemas from demand and supply chains are the same as explained for conventional DWs (Figure 5.11). A new phase of adding spatial support for resulting multidimensional schema is included (the lower line in the figure). Notice that similar to the previous approaches, external sources may be needed for obtaining spatial data.

To represent each of the conceptual schemas (initial and final in every approach), the conceptual model for spatial DWs as proposed in Chapter 3 of this thesis can be used. Notice that we did not present here examples for deriving conceptual models with spatial elements, since the proposed methodology only slightly differs from previously-explained proposal for the conventional DW design and Chapter 3 already includes examples of spatial DWs.

The specification of requirements and conceptual modeling of spatial DWs demand designers to be experts in spatial data management as well as in the concepts related to spatial DWs as the ones explained and exemplified in Chapter 3 of this thesis. Designers must also be able to work with users having different knowledge about spatial data, i.e., the ones that are experts and the others than are just beginning the interest of exploiting the advantages that spatial data representation can give for supporting decision-making process.

### 5.8.1.3 Logical and physical design

Logical and physical design for spatial DWs should consider aspects mentioned for the conventional DWs (Sections 5.6 and 5.7).

In Chapter 3 we already specified mapping rules from our conceptual model into the object-relational model. We also showed several examples of applying this mapping for spatial levels, spatial hierarchies, spatial fact relationships, and spatial measures.

In the mentioned chapter we also referred to the physical level considerations using Oracle 10g as an example of a DBMS. We presented our rationale for using object-relational database management systems with spatial extensions. We also referred to different physical models that can be used for storing spatial data, i.e., topological or
5.8. A METHODOLOGY FOR SPATIAL AND TEMPORAL DWS

Figure 5.14: General phases in the demand/supply-driven approach for spatial DWs.

We mention the necessity to make the decision whether vector or raster data representation is preferable. All these aspects should be considered when spatial DWs are implemented.

Other features should also be taken into account when designing spatial DWs. For example, spatially-enabled DBMSs usually include special kinds of indices that allow to improve performance during execution of queries with spatial data. Two kinds of indices are commonly used, i.e., R-tree and quadtree. Other facilities that help in developing and executing ETL processes should be also considered, for example, extraction of spatial data from different kinds of source systems, transformations from raster to vector representation or vice versa, loading mechanisms for spatial data, among others.

5.8.2 Temporal extension

5.8.2.1 Temporal support for elements of multidimensional models

In our multidimensional model temporal support may be given for levels, attributes, hierarchy links, and measures. Different temporal support may be considered for each multidimensional element. For example, levels can include lifespan and transaction
CHAPTER 5. METHODOLOGY

time. Level's attributes, hierarchy links, and measures can have valid time and transaction time. Lifespan, valid time, and transaction time for all elements specified above should come from source systems as already explained in Section 4.3. Additionally, all elements may include data warehouse loading time that is generated in the DW.

5.8.2.2 Requirements specification and conceptual design

Similar to the previous case of methodology for spatial DW design, we consider three different methods for the requirements specification and the creation of conceptual schemas. We also distinguish different cases considering whether users are familiar with the concepts related to temporal data and whether the source systems are temporal or temporal support is not provided at all. Notice that even though temporal DBMSs are not yet available, we suppose the existence of source systems that may include some temporal support and manage it in an empiric way.

The modifications for each method of requirements specifications will be similar as the ones already explained for spatial DWs (Figure 5.12, 5.13, and 5.14) with the difference that instead of adding spatial support, temporal support will be considered. Next, we briefly describe each of these approaches to avoid ambiguity when applying them.

Demand-driven approach When users are able to express their needs for having temporal support for different elements of multidimensional schemas, the phases for the demand-driven approach for the temporal DW design remain the same as for the conventional DW design (Figure 5.3). However, each phase of this method must refer additionally to the required temporal data. The phase of checking data availability considers users' demands related to non-temporal as well as temporal elements. In Section 4.3 we already indicated which temporal support can be obtained taking into account different kinds of source systems. However, if users' requirements for temporal data cannot be satisfied, the decision should be made whether to modify the source systems to be able track changes to data or to prepare an adequate intermediate storage (e.g., staging area) and processes to capture changes in DWs.

On the other hand, users (and/or designers) may prefer to first refer to the analysis needs without considering temporal data and later on, include the temporal support to expand analysis spectrum as already exemplified in Chapter 4. In this case, an additional phase of adding temporal support is included similar to the spatial DW design (the lower part in Figure 5.12).

The inclusion of temporal data should consider different temporal types and multidimensional elements. First, we focus on the lifespan. Therefore, every one-level dimension or every level forming hierarchies is verified for inclusion of lifespan. Similar analysis is conducted for levels' attributes and links between hierarchy levels making a decision whether valid time should be associated with these elements. If the users require to analyze the sequence of changes occurred to source data and/or when the data was introduced into the DW, the transaction time and/or data warehouse loading time, respectively, should be included for each element that represents users' interest.

A different approach should be taken for measures. Currently measures are handled
as attributes using a time dimension for indicating their valid time\textsuperscript{12}. The members of the mentioned dimension representing dates should be transformed into temporal support for measures\textsuperscript{13}. However, users may be not aware of the possibility and advantages of including transaction time from source systems and data warehouse loading time generated in a DW. Therefore, designers should be able to recognize from users’ requirements that the inclusion of these temporal types for measures could help in the analysis process.

Similar to the previous case, during the phase of checking data availability, designers should verify whether this temporal support can be implemented based on the existing source systems or some additional structures should be created to capture the future data changes.

**Supply-driven approach** This approach will consider whether temporal support is included in source systems in a similar way as was explained for spatial data. Here also a special derivation process is needed and to our knowledge there are no proposals for such a process. Therefore, manual derivation based on designer skills may be used instead. However, since this can be a complex task, we recommend to derive first a conceptual schema without the temporal extension. After users determine which elements should be left for analysis purposes, an additional analysis is conducted to specify the required temporal support. The proposed phases are shown in Figure 5.13 (the lower part) replacing the phase of adding spatial support for adding temporal support. Similar to the demand-driven approach above but now during the phase of mapping refinement, designers must check whether temporal data required by users is available. In the negative case, the decision of how to capture changes to data as explained for the previous approach should be taken.

**Demand/supply-driven approach** Even though the phases and descriptions given for the demand/supply-driven approach for the spatial DW design apply for the temporal DW design, we propose to ignore the temporal aspect during the design phases in each chain, i.e., to follow the same procedure as explained for the first four phases for conventional DWs (Figure 5.11). Then, after successful matching between schemas obtained from the demand and the supply chains, users’ demands related to temporal support should be considered, i.e., the phases as specified for the spatial DW design (the upper line in Figure 5.14) could be used considering now temporal support instead of spatial one.

Temporally-extended conceptual schemas obtained from each approach specified above can be represented as proposed in Chapter 4 of this thesis. We did not give here examples, since in the mentioned chapter several multidimensional models with temporal support for dimensions and measures were included.

The process of requirements specification and conceptual modeling for temporally-extended multidimensional schemas requires designers with good knowledge related to temporal semantics of DWs, i.e., they should be able to understand different temporal

\textsuperscript{12}This is a most common temporal support currently used in DWs.

\textsuperscript{13}For more details refer to Section 4.8.
types and implication of having them in the DW context. They should also be capable to explain very technical concepts of time-varying multidimensional elements, such as the possibility to represent dimension data changes, the inclusion of different temporal support for measures, management of different time granularity, among others, to users who probably would not know about them.

5.8.2.3 Logical and physical design
The logical level design is obtained using mapping rules. Whether the relational model or the object-relational model is used, different mapping rules should be specified. For the former, the recommendations given by Gregersen et al. [66] or Snodgrass [190] can be applied. For the object-relational model, the rules specified in Section 4.10 can give useful insights for designers and implementers. Throughout Chapter 4 we presented examples that applied these rules using the SQL:2003 standard and Oracle 10g.

However, during the previous phase of the requirements specification a situation of the lack of temporal support in source systems may be revealed. Therefore, according to the decision made in the previous phase, additional ETL processes and new structures for source systems or for staging area should be created. This in order to capture the source data changes that currently are not available and are required by users for analysis.

Physical level design concerns ensuring efficient execution of queries and facilitating management of temporal DWs. Therefore, the considerations specified for conventional DW design should also be included here, i.e., indexing, materialized views, fragmentations, among others. Specially, the fragmentation techniques may improve performance if the adequate criteria for partitioning of dimension and fact tables are used. For example, the solutions presented by Martín and Abelló [127] as described in Section 4.11 could be considered.

5.9 Related work
In the following, we group the related work in different areas to facilitate the reading. Notice that several works, for example, [100] were already described in Section 5.2 in the introduction to currently existing approaches.

5.9.1 Overall process
The work of Golfarelli and Rizzi [61] presents a DW methodology with the following steps: (1) analysis of the information system, (2) requirements specification, (3) conceptual design (based on [60]), (4) workload refinement and schema validation, (5) logical design, and (6) physical design. Even though, it corresponds to the traditional database design, it includes an additional phase of workload refinement, which allows to determine the expected data volumes.

---

14See Chapter 4 for more details.
15See description in Section 4.11.
5.9. RELATED WORK

Luján-Mora and Trujillo [114] present a methodology for DW design using the UML notations. Their proposal allows to deal with all DW design phases from the operational data sources to the final implementation including the ETL processes. First, they discuss which schemas and mappings between them should be included in the DW design. They distinguish schemas for operational data, for the DW conceptual design, for the DW storage, and for the so-called business model. The latter defines different ways of accessing the DW data from the final users' viewpoint. Two mappings are included: one for the ETL process that establishes relations and transformations between operational and DW data, and another one for exportation processes when data must be mapped from the DW conceptual into the logical schemas. Using these elements, authors define different activities and group them in four categories: analysis, design, implementation, and test, i.e., skipping the logical-level design.

Chenoweth et al. [34] propose a methodology for developing data marts. They include the following phases: requirement analysis, logical modeling, selection of the DBMS, mapping the logical model to a DBMS, development and evaluation of the physical model, system tuning, and system implementation/monitoring. The conceptual design is missing in this methodology. The content of each phase is then contrasted with its corresponding phase in the operational (relational) database development process highlighting the differences, such as the requirement analysis based on the business process model or the logical model based on the star schema.

Jarke et al. [89] propose the DWQ design methodology for DWs. This methodology consists of six steps. Steps 1 and 2 refer to enterprise and source conceptual model constructions. Step 3 is responsible for the definition of required aggregated for the specific clients. Then, in step 4 the translation of aggregates into OLAP queries takes place. Step 5 specifies which of the aggregated data should be materialized for improving system performance. Finally, the transformation and integration of data in source system are considered in step 6. The proposed methodology does not follow the traditional database design phases and focuses more on the issues related to particularities of OLAP systems and to the improvement of systems performance.

Pereira and Becker [164] as well as Carneiro and Brayner [30] consider that even though many enterprises could benefit from the use of DWs, they lack within their own personnel the experienced staff required for development of a DW and may not be able to make use of an outside development team due to several reasons, such as costs, privacy, deadline, or availability. Therefore, they propose a new methodology for the development of DW pilot projects that can be used by professionals experienced in the development of traditional databases. The approach of Pereira and Becker [164] is based on the Kimball et al. life-cycle [100] and includes several additional phases, such as experimentation and prototyping. On the other hand, Carneiro and Brayner [30] first build a prototype and then, based on the spiral model, include several iterations building a more complete and a new DW version in each iteration. The idea of inclusion of a pilot project in the DW lifecycle allows to introduce experience into the internal team in constructing a DW; it also reduces the risk of failure.

Other authors, for example, [82, 169, 183], briefly refer to the DW design phases which correspond to database design phases. These phases are general enough to fit and facilitate the tailoring to any DW projects. However, they lack more detailed guidelines.
5.9.2 Requirements specifications

As was mentioned in Section 5.2, different approaches are used for the requirements specification phase. In this section we describe them in more detail grouping them in data-driven, user-driven, business (goal)-driven, and demand/supply-driven approaches. We do not present critics of each method since a more general description of advantages and disadvantages of each of them was described in Section 5.5.

Data-driven Böehnlein and Ulbrich-vom Ende [20] propose a technique for the derivation of initial logical DW structures from the conceptual models of the underlying operational source systems. They assume three stages. The first stage of defining measures is based on the chain goals - services - measures. After determining measures, the derivation of dimensions is realized. This derivation is based on existency dependencies that are visualized using SERM (Structured Entity-Relationship Model). Some hierarchies may also be derived considering cardinalities existing between different entity types. The final step consists of specifying integrity constraints along the dimension hierarchies. Since the process of defining hierarchies is not always successful and requires creativity and considerable knowledge of the application domain, they consider that the derivation of multidimensional models should not be automated.

Golfarelli et al. [60] present a graphical conceptual model for DWs called the Dimensional Fact Model and propose a semi-automated process to build it from operational ER schemas. The first phase requires collecting the documentation of the underlying operational systems. Based on this documentation the designer and end-users determine facts and the required workload. Then, a semi-automated process is applied to find other multidimensional elements. For each fact, an attributes tree is built. The next two steps of pruning and graphing the attributes tree allow to exclude attributes not required for analysis purposes. Finally, the navigation through the schema along x-to-one relationships allows to determine dimensions and their hierarchies.

Moody and Kortnik [138] propose a method for developing dimensional models from enterprise data models represented using the ER model. The four-steps approach starts from classifying entities to represent facts with associated measures, dimensions, and hierarchies. For the latter they consider one-to-many relationships between the functionally-dependent entities. Then, different multidimensional schemas can be created, such as star or snowflake schemas. The final step includes evaluation and refinement of the proposed schema to produce a final data mart with a more simple structure.

Cabbibo and Torlone [28] present a design method that starts from an existing ER schema. They assume that this schema describes a “primitive” DW containing all the operational information that can support the business processing, but not yet tailored to analysis activity. They derive a multidimensional schema and provide implementation in terms of relational tables as well as multidimensional arrays. The derivation of the multidimensional schema is structured into several steps. In the first step, facts along with their measures have to be selected, and afterwards, dimensions for a fact are identified by navigating the schema. Then, in the second step the initial ER schema is restructured in order to express facts and dimensions explicitly, thus arriving at a multidimensional representation.
Hüsemann et al. [82] use the operational ER schemas of transactional applications that deliver basic information to determine eligible multidimensional requirements. Business domain expert select strategically relevant operational attributes that are classified as dimensions or measures. The resulting requirements are presented in a tabular list of attributes along with their multidimensional purpose. Supplementary information (integrity constraints, additional derived attributes) can be added informally in a textual appendix. Then based on these information, they focus on conceptual design based on functional and generalized multidimensional dependencies [110].

One of the well-known practitioners, Inmon [86] also considers that the DW environment is data driven in comparison to classical systems, which are requirement driven. According to the author, the requirements should be understood after a DW is populated with data and being used by the decision support analyst. He derives the data model by transferring the corporate data model into a DW schema and by adding performance factors.

User-driven Paim et al. [152] propose a phase-oriented methodology for requirements analysis of DW systems consisting in requirements management planning, requirements specification, and requirements validation. The phase of requirements specification includes requirements elicitation based on communication with stakeholders. Different techniques are proposed, such as interviews, prototyping, and scenarios. This phase requires the presence of application domain users. Paim and Castro [153] extend the methodology present by Paim et al. [152] calling it DWARF (Data Warehouse Requirement Definition) by inclusion of non-functional requirements, such as performance or accessibility.

Trujillo et al. [198] present a conceptual model based on the UML diagrams for designing DWs. This model is then used to create a conceptual schema within the methodology proposed for DW design [114]. This methodology includes several steps that are user-driven, such as requirements specification, conceptual modeling, definition of data marts.

Freitas et al. [58] consider that to understand correctly facts, dimensions, and relationships existing between them, an effective user participation is required. They present MD2, a tool based on the dimensional data modeling approach, which facilitates the user participation in several steps involved in the development of DW applications.

Imhoff et al. [84] propose the methodology that first develop the so-called subject data model. The latter represents subject of analysis using the ER model. This subject area model can be developed based on users' requirements obtained applying different techniques, such as interviews or facilitated sessions.

Business (goal)-driven Böehnlein and Ulbrich-vom Ende [21] derive DW structures from business process model considering that users often are not able to formulate their demands clearly. They consider that a DW is designed according to a set of relevant business subjects and each of these subjects centers around business process. Based on the Semantic Object Model (SOM) methodology used for business engineering, they derive initial DW structures. The SOM methodology helps them to obtain the so-called Conceptual Object Schema (COS), which is later on used for identifying facts,
measures, and dimensions. Since they do not use any conceptual multidimensional model, they present a logical model based on the star schema.

Bruckner et al. [25] introduce three different abstraction levels for DW requirements considering business, users, and system requirements. Business requirements represent high-level objectives of the organization identifying the primary benefits that the DW system will provide to the organization. User requirements describe the task that the users must be able to accomplish having a DW system. These requirements have to be in concordance with business requirements. Finally, detailed system requirements refer to data, functional, and other kinds of requirements.

Mazon et al. [129] present an approach for including business goals in DW requirement analysis. Then, these requirements are transformed into a multidimensional model. They use i* technique [216] that helps to understand the organizational environment and goals. This technique requires to build two kinds of models, i.e., a strategic dependency model, which describes the dependency between actors in an organizational context, and a strategic rationale model, which is used for understanding actor interests and the way they might be addressed. They adapt these models to the DW context giving specific guidelines for building them. Then, others guidelines are added for the process of mapping the models to the conceptual multidimensional model based on the UML notation.

Giorgini et al. [59] propose a goal-oriented approach for DW requirements analysis based on the Tropos methodology. The latter is an agent-oriented software development methodology, which also applied i* conceptual framework [216]. Their starting point is the supposition that an explicit goal model of organization is given. Goals are considered at different levels of management leading to creation of the so-called organizational and decisional models. These models are refined in order to approach to definition of facts, measures, and attributes. The highly-technical requirement models are then mapped to conceptual multidimensional model of Golfarelli et al. [60].

Prakash and Gosain [169] propose a requirements elicitation process for DWs. The process starts with the determination of the goals of an organization. Secondly, the decision-making needs are specified, and finally, the information needed to cover these decision, i.e., the source data, is identified.

List et al. [113] apply the UML use-cases for the phase of the requirements specification. They focus on business processes to identify multidimensional objects that correspond to facts, measures, and dimensions.

Kimball et al. [100, 101] base their DW development strategy on choosing core business processes to model. Then, business users are interviewed to introduce the DW team to the company’s goals and to understand users’ expectations of having the DW. Even though this approach lacks formality, it was proven to be successful in many DW projects.

Combination Bonifati et al. [22] present a method for the identification and design of data marts. This method consists of three general parts: top-down analysis, bottom-up analysis, and integration. Top-down analysis emphasizes the user’s requirements and it requires a precise identification and formulation of goals. The authors use the so-called Goal/Question/Metric methods [9], which includes not only goals specification but also their characterization. Based on that, a set of ideal star-schemas is created.
5.9. RELATED WORK

On the other hand, the bottom-up analysis aims at identifying all the star schemas that can be realistically implemented on the top of the available source systems. This analysis requires source systems to be represented using the ER model. After several steps of the proposed algorithm, the graph-like structure of the star/snowflake schemas is elaborated. The final phase of integration is decomposed in several steps and it allows to match the ideal star-schemas with realistic ones based on the existing data.

Phipps et al. [165] describe the process of creation of a DW conceptual schema using first the data-driven approach. The proposed automated technique allows to obtain candidate DW conceptual schemas based on the existing data. They consider that relying on only data sources may lead to the situation that incomplete knowledge of user needs will be satisfied and not adequate DW schemas may be produced. Therefore, they expand the methodology by the inclusion of user-driven approach. User needs are represented as queries that users wish to perform. Then the process of revision whether the schemas obtained using data-driven approach can answer user queries is proposed.

Winter and Strauch [213] consider the support for information requirements analysis as a very important aspect of project management during the DW system development. The proposed methodology first determines information requirements of DW users and then, compare them with aggregate data from source systems. The non-covered information requirements are homogenized and prioritized. Then, the selection of more detailed data from source systems is considered. The resulting multidimensional schema is evaluated. If the user requirements are still not met, other iterations between different phases begin.

5.9.3 Conceptual, logical, and physical level design

Work related to conceptual as well as logical models for design of DWs was already described in Section 2.6. Moreover, many proposals for requirements specifications include the creation of conceptual or logical DW models as described above.

Very few work refers to physical aspects for implementing DWs. Peralta and Ruggia [163] describe general design guidelines for materialization of aggregates, horizontal fragmentation of facts, and vertical fragmentation of dimensions. The necessity to consider the fragmentations is also mentioned in other works (e.g., [62, 97, 163]). On the other hand, Karayannis et al. [97] propose an abstract processing plan that allows to evaluate and to optimize the execution of star queries over fact tables that are clustered according to the dimension hierarchies. Many works propose specific algorithms for creating and managing precomputed aggregates or materialized views [68, 182, 191]. Several works from Stanford University DW project [192] belong also to this group.

Well-known commercial DBMSs, such as Oracle, Sybase, IBM, Microsoft usually provide a product-specific documentation that explains physical features which can facilitate the DW implementation and management as well as improve query performance. These features refer to different aspects, such as mentioned in the section of physical design (Section 5.7).

In general we can say that despite of the wide diffusion of the DW technology and concepts, the DW developer teams still lack a methodology that helps and guides them in the different stages of the DW design. Most of the existing works refer to one or
two specific phases required to build a DW, for example, to requirement specification, to conceptual or logical modeling, or to physical aspects that allows to improve system performance. Even though some works propose the methodology for developing DWs, they either skip some of the phases used in the conventional database design or include new ones specific to the problem or to the model presented by the authors. Therefore, the provided solutions cannot be used as a general framework that can be tailored according to the specific needs of DW projects.

5.9.4 Spatial and temporal design methodologies

Section 3.8 was already devoted to conceptual and logical models for spatial DWs as well as for spatial database design. Since spatial DWs represent a new research area, to our knowledge there are not works related to the design methodology. On the other hand, even though spatial databases are widely used and investigated, very few works refer to the methodology for their design.

Rigaux et al. [178] do not refer explicitly to methodology for spatial database design, however, they include examples using a conceptual schema based on the OMT formalism. Then, the conceptual schemas are mapped to the relational and to the object-oriented models. Since the conceptual representation does not allow to include spatial data, they use spatial abstract data types to represent such data in the logical level. For each spatial attribute corresponding abstract data type is used. Finally, they refer to physical aspects, explaining different features, such as space-partitioning algorithms, implementation of topological operations, spatial access methods, spatial join operations, among others.

Shekar and Chalwa [186] also refer to the three-step design methodology for spatial databases. First, they use the ER model to represent spatial as well as non-spatial data. For the former, they use multivalued attributes, which later are mapped to separate relational tables and handled as non-spatial data. Additionally, they extend the ER model with pictograms to represent spatial data, which allows them to reduce the number of elements in the ER schema. This also simplifies the relational schema resulting from mapping. Then, different physical aspects are discussed, such as spatial indexing techniques, file structures, clustering, spatial join, among others.

Worboys and Duckham [214] explicitly present conceptual and logical design as well as mapping between them. For representing conceptual schemas they use either the extended ER model or the UML notation. Then, the ER schemas are mapped to relational databases while the UML schemas to object-oriented databases. Similar to the above-mentioned works, they refer to several physical features required for representing and handling spatial data in DBMSs.

Parent et al. [160] propose MADS a conceptual model for designing spatio-temporal databases. They also refer to the logical representation including a two-step process. In the first step, the MADS constructs are transformed to more simple representations. In the second step, a dedicated wrapper is built that allows to reformulate the MADS specifications to the language of the target GIS or DBMS.

Regarding to temporal DWs, in Section 4.11 we already referred to existing conceptual and logical models for temporal DWs and temporal databases. We also mentioned about two existing approaches for temporal database design, i.e., based on the
(temporal) normal form and on the temporally-extended conceptual models and their mappings to a relational model.

Very few works propose a comprehensive methodology for temporal database design. Snodgrass [190] refers to a traditional database design phases and applies them for temporal database design. Initially, all temporal aspects are ignored when developing a conceptual schema, i.e., changes in data are not captured at this stage. In this way existing conceptual design methodologies can be applied. Only after full design is complete, the conceptual schema is augmented with the time-varying elements. Similarly, the logical design is developed in two stages. First, the non-temporal schema is mapped into the non-temporal relational schema using the already existing mapping strategies, for example, [48]. In the second stage of logical design, each of the temporal element is included either as part of the tables or as integrity constraints.

Detienne and Hainaut [41] also propose a methodology that is similar to the conventional one, i.e., it includes the conceptual, logical, and physical phases. During the conceptual design, first a non-temporal conceptual schema is built. Then, designers choose which elements of the schema should have temporal support and which temporal types are required. Finally, the schema is normalized according to the consistency rules defined by them. The temporal logical design phase consist in translating the conceptual schema into the relational one and in adding attributes that represent the required temporal support. The physical design phase considers aspects related to table partitioning, indexing, and creating auxiliary structure to improve system performance. In all phases designers are assisted by a CASE tool that makes the process of developing a temporal database easier and more reliable.

Nevertheless, similar to spatial data warehouse design, there is a lack of works related to methodology for temporal DW design.

### 5.10 Summary

In this chapter we presented a general methodological framework for conventional, spatial, and temporal data warehouse design. First, we modified the DW lifecycle proposed by Kimball et al. [100] by the inclusion of the conceptual design phase and by expanding the activities required for the logical and physical phases.

For the requirements specification phase we proposed three different methods. The demand-driven method focuses on business or user needs. Those needs must be aligned with the goals of the organization in order to ensure that the DW system can provide the necessary support for decision processes at all organizational levels. However, the final DW model includes only those users' requirements, for which corresponding source data exist. The second approach, supply driven, develops the DW model based on the structures of the underlying operational databases, which are usually represented using the ER or the relational model. Users' needs are considered only in the scope of available operational data. The third approach, demand/supply driven, combines both approaches. It matches user's needs with data availability.

The requirements specification phase is refined until elements of the multidimensional model can be distinguished, such as facts, measures, dimensions, and hierarchies. Then, the conceptual schema can be build and evaluated by users leading to its final
The next phases of our methodology correspond to the classical database design. Therefore, the mapping to the previously-chosen logical model is specified and physical structures are defined. We additionally expanded these phases with the refinement of the ETL processes and design of the staging area, if it is necessary. Notice that the physical-level design should consider specific features of DBMSs related to the particularities of DW applications.

We also referred to the methodology for spatial and temporal DW design. Based on the methodological framework for conventional DW design, we modified some phases and included additional ones depending on users' knowledge about spatial (temporal) data and on whether source systems provide spatial (respectively, temporal) support.

The proposed methodology has several advantages. First, it does not depend on conceptual model, on target implementation platform, or on database logical and physical organizations. It also refers to the overall lifecycle of DW development including the conceptual modeling phase ignored by most practitioners and by some scientific works. Additionally, it classifies and organizes the variety of currently existing approaches for requirements specifications into a coherent whole. It generalizes the proposed approaches to make them easy to understand without loosing their semantic differences and it provides clear guidelines stating in which situations each method may give better results. Therefore, it allows designers to choose an adequate approach according to the particularities of a DW project.

Moreover, including a part of the ETL processes (i.e., mappings and transformations) in the conceptual-level design and extending the specifications of these processes in the logical-level design allow to define and refine such processes without being burdened with implementation details of the underlying database management systems. The subsequent phase of physical design will give the necessary tools for the implementation of these processes.

Finally, the extension of methodology for the inclusion of spatial and temporal support allows the users to express their needs for having spatial and time-varying data. The specification of required phases depending on whether source systems contain spatial or time-varying data guides designers and implementers indicating them when and how spatial or temporal support can be included.
Chapter 6

Conclusions and future work

6.1 Conclusions

Nowadays many organizations use Data Warehouse (DW) and On-Line Analytical Processing (OLAP) systems to support the decision-making process. In these systems, it is recognized that a multidimensional model is well suited for expressing users' requirements concerning the focus of analysis. A multidimensional model includes measures representing topics of analysis, dimensions used to narrow this analysis, and hierarchies giving the possibility to choose different levels of detail. A hierarchy represents some organizational, geographic, or other type of structure. They are important since OLAP systems automatically aggregate the measures while traversing hierarchies.

Nevertheless, several important issues need still to be addressed with respect to multidimensional models. Firstly, there is not a commonly-agreed conceptual model for representing multidimensional data. Secondly, existing conceptual multidimensional models do not provide logical representations, leaving to implementers to realize the mapping from a conceptual to a logical schemas. Thirdly, many kinds of complex hierarchies arising in real-world situations are not addressed by current DW and OLAP systems. Fourthly, current multidimensional models do not include spatial data even though this kind of data can considerably improve analysis. Finally, the traditional time dimension allows only to represent changes to measures leaving to implementers the responsibility to manage changes to dimension data.

Therefore, in response to these requirements, in this thesis we proposed the MultiDimER model - a conceptual model able to express data requirements for DW and OLAP applications including extensions for managing spatial and temporal data.

Chapter 2 defined the MultiDimER model comprising fact relationships, measures, dimensions, and hierarchies. For the latter, we took a conceptual approach and studied in a systematic way the different kinds of hierarchies existing in real-world applications and scientific works related to multidimensional modeling. Based on these studies, we proposed a conceptual classification of such hierarchies and graphical notations for them. The notations allow a clear distinction of each type of hierarchy taking into account their differences at the schema as well as at the instance levels.

Further, we provided a mapping of the constructs of the MultiDimER model to the relational model. Since our model is based on the ER model, we used a traditional approach for this mapping. In order to better represent at a logical level the
semantics of generalized hierarchies, we proposed specific mapping rules for them. We also discussed current approaches for logical representation of some types of hierarchies comparing them and stating in which situations the different mappings work better.

Chapter 3 extended the MultiDimER model with spatial data. We proposed the inclusion of spatial dimensions with spatial hierarchies, spatial fact relationships, and spatial measures. We referred to the previous classification of hierarchies and showed that it is also applicable for spatial hierarchies. We pointed out that the summarizability problem may occur due to the different topological relationships existing between hierarchy levels. We classified these relationships according to the complexity required for developing procedures for measure aggregations. Moreover, we showed that when a model includes more than one spatial dimension, a topological relationship between them is required. We also extended the analysis to spatial measures and presented two cases: (1) when a spatial measure is represented by a geometry, and (2) when a spatial measure is the result of a calculation using spatial or topological operators. We discussed the necessity of having spatial aggregation functions defined for spatial measures when hierarchies are presented in the model. Moreover, we relaxed the requirement of having a spatial dimension to represent a spatial measure, i.e., we allow to include in the model spatial measures without the presence of spatial dimensions.

Finally, we proposed mappings to the object-relational model based on the SQL:2003 standard along with the examples using a commercial system Oracle 10g Spatial. However, even though the mapping to logical level is based on well-known rules, it does not completely represent the semantics of the MultiDimER schema. To ensure the semantic equivalence between conceptual and logical schemas during the transformation process, integrity constraints were implemented mainly using triggers.

Chapter 4 was dedicated to the temporal extension of the MultiDimER model. We proposed the inclusion of different temporal types, such as valid and transaction time, which are obtained from source systems, in addition to the data warehouse loading time generated in a temporal DW. We included this temporal support for levels, hierarchies, and measures. We allowed levels to include time-varying attributes and/or lifespan support. For hierarchies, we discussed different cases whether changes to levels or to links between them are required to be kept. By means of real-world examples, we showed the necessity to consider different temporal types for measures. We also referred to the problem of multiple time granularities for measures in source systems and DWs. We proposed temporal types for aggregated measures and discussed the necessity to perform adequate transformations for time granules and measures. Finally, we included two mappings, to the ER model and to an object-relational model based on the SQL:2003 standard.

Chapter 5 described the design methodology for conventional, spatial, and temporal DWs. We proposed several phases that correspond to those of the design methodology for traditional databases, adapting them to the DW context. For the phase of requirements specification, we proposed three different approaches that use as a driving force, respectively, business requirements, available data in source systems, or a combination of both. We gave guidelines for DW developer teams that allow them to choose an
option that better suits to the specific DW project. We also briefly referred to the logical-level design including a new part of the ETL process. Finally, we proposed several recommendations for physical design and gave examples of their implementation using Oracle 10g.

Appendix A, B, and C presented the formalization of the proposed model including spatial and temporal extensions. This formalization is based on denotational semantics and contains the specifications of syntax, semantics, and associated constraints. Appendix D includes the specification of notations used in this thesis for representing constructs of the entity-relational, relational, and object-relational models as well as of the conventional, spatial, and temporal data warehouses.

The proposed conceptual model and its mapping to a logical representation benefit users, designers, and implementers of DW and OLAP systems and applications. It allows designers to properly represent data requirements improving the communication with decision-making users. Using the MultiDimER model, they are able to capture in a better way the particular semantics of DW and OLAP applications than using a logical or the conventional ER or UML models. Most of the existing conceptual multidimensional models do not distinguish between the different kinds of hierarchies proposed in this thesis, although some of these models can be extended to take into account the proposed hierarchy classification. Further, a conceptual representation of hierarchies offers a common vision of these hierarchies for DW and OLAP systems implementers; it provides the requirements that allow to extend the functionality of the current OLAP tools.

Moreover, the proposed mappings of different types of hierarchies to the relational model along with the analysis in which situations different mappings work better, can guide implementers for the physical design of the DW. Currently, since there is a lack of the logical level representation for different kinds of hierarchies, designers must apply different “tricks” at the implementation level to transform some hierarchy types into simpler ones.

The spatial extension of the MultiDimER model aims at improving the data analysis and design for spatial DW and spatial OLAP applications by integrating spatial components in a multidimensional model. In this way, decision-making users can represent in an abstract manner their analysis needs without considering complex implementation issues. Further, spatial OLAP tools developers can have a common vision of the different features that comprise a spatial multidimensional model and of the different roles that each element of this model plays. Mapping to an object-relational model shows the feasibility of implementing spatial DWs in current commercial database management systems. However, even though the mapping to logical level is based on well-known rules, it does not completely represent the semantics expressed in the conceptual level. Therefore, additional programming effort is required to ensure the equivalence between the conceptual and the logical schemas.

The temporal extension of the MultiDimER model allows to include temporal semantics as an integral part of DWs. It also allows to expand the analysis spectrum for decision-making users. Since it is a platform-independent model, it can be transformed either to other conceptual models or mapped to logical models. We showed that schemas represented using our model are less complex and require less knowledge about technical aspects than schemas transformed into the ER model. Further, even though
the object-relational model better represents temporal semantics than the relational model by allowing to group data with its corresponding timestamps, it is not always adequate for all time-varying elements of temporal DWs, for example, for measures.

Proposing the methodology for conventional, spatial, and temporal data warehouse design facilitates the development of DWs. Since it is model and software independent, it can be used as a general framework for DW development in different areas of human activities. It gives benefits to the DW developer team providing a systematic specification of different approaches that can be used for the requirements gathering phase. The guidelines given for each of the approaches allow to choose one according to the knowledge and experience of the developer team as well as to the particularities of the DW project. It also helps users, since designers can choose an approach that fits better according to users' time constraints for participating in the DW project, their identification with the business goals, and their motivation for having a DW. Further, implementers can profit significantly from the specification of a DW structure and ETL processes developed during the conceptual and logical phases. Since the proposed methodology combines different approaches, it creates a common base to cope with complex issues joining the effort of scientists and practitioners. It also refers to new aspects considering the inclusion of spatial and temporal support on each phase of the methodology.

Finally, the presented syntax and semantics of the model gives the necessary essentials for its future implementation.

6.2 Future work

The work reported in this thesis may be continued in several directions in the future. We present different areas in which the extensions may be realized.

6.2.1 Conventional data warehouses

The conceptual model presented in this thesis is based on the ER constructs with its corresponding mappings to relational or object-relational databases. Since the target systems are well-known implementation platforms with clearly defined structures and operations, our model does not require the specification of operations. The set of operations available in the target systems can be used to implement more specific operations, if required. However, an interesting issue would be to propose the aggregation procedures for different kinds of hierarchies, for which summarizability conditions do not hold, i.e., for asymmetric, generalized, non-covering, and non-strict hierarchies. Some solutions for asymmetric, non-covering, and non-strict hierarchies are proposed by Pedersen et al. [161]. Their approach transforms these hierarchies into symmetric, covering, and strict, respectively. Nevertheless, for the latter, the additional structures are created that may not correspond to users' needs. Some insights for developing aggregation procedures can be obtained from works of [80, 142, 168].

Another aspect is the inclusion of other features of the ER model. Currently, the MultiDimER model uses entity types, attributes, and relationship types with their usual semantics. Additionally, our model offers explicit support for representing dif-
6.2. FUTURE WORK

Different kinds of hierarchies. However, we do not consider other ER constructs, such as weak entity types, generalization, multivalued attributes, and composite attributes. The inclusion of these features is not straightforward and requires analysis of their usefulness in multidimensional modeling. Some ideas can be taken from [6] where the problem of representing generalization/specialization hierarchies in a multidimensional model is discussed or from [3] where a drill-across operation is proposed for dimensions represented by generalization/specialization relationships.

Further, even though multidimensional storage in MOLAP systems is vendor-specific, it would be an interesting topic to investigate how the proposed hierarchies can be mapped to an array-based physical structure. The analysis of difficulties in representing these hierarchies will give insights to their future implementation on current MOLAP platforms. There are different models that can be used as a starting point, e.g., [28, 205]. References to these and other models are included in [206].

6.2.2 Spatial data warehouses

The spatially-extended MultiDimER model allows to include spatial measures and spatial dimensions. The distinction between them is important since they play different roles. Considering that dimensions may comprise spatial hierarchies, we already discussed the impact that topological relationships between hierarchy levels have on aggregation procedures. However, it is still necessary to extend this analysis taking into account different topological relationships that may exist between spatial measures. Some solutions for handling this problem can be taken from [162]. However, they apply them for spatial hierarchies while in our approach spatial measures should be considered.

Another aspect for spatial DWs is their extension to manage 3-dimensional objects. Coping with these kinds of objects is necessary in many application domains, such as urban planning processes, telecommunications planning, or disaster management [67]. In the literature several conceptual models for 3-dimensional objects have been proposed (e.g., [103, 109, 204]). Similar to 2-dimensional objects, models that include 3-dimensional objects consider not only geometry but also topological aspects [220]. The extension of the MultiDimER model should first consider the analysis of different kinds of applications that use 3-dimensional objects. This will help to clarify whether the multidimensional view of data would help in the decision making process for these kinds of applications. Then, the concepts proposed in this thesis for the spatial extension of the MultiDimER model should be revised, for example, the classification of topological relationships between spatial levels forming a hierarchy for aggregation purposes or aggregations of spatial measures.

Further, as explained in Section 3.1, our model refers to a discrete (or object) view of spatial data. However, different phenomena can be represented using a continuous (or field) view as briefly mentioned in Section 3.1, for example, temperature, altitude, soil cover, pollution. Both kinds of views are important for spatial applications [160]. Therefore, the MultiDimER model can be extended by the inclusion of field data. Different aspects should be discussed, for example, the representation of the field data in the model, spatial hierarchies formed by levels represented by field data, spatial measures representing continuous phenomena and their aggregation, fact relationships
CHAPTER 6. CONCLUSIONS AND FUTURE WORK

that include spatial dimensions with field data, among others. In the spatial database community, there are several approaches to the mentioned problems. For example, [160] use different representations for object and field data in a conceptual spatio-temporal model called MADS, [85] present algorithms that can be used for creating spatial hierarchies based on field data, Tomlin's algebra [196] or advanced map operations [50] can be used for analysis of how to manage spatial field measures represented by field data.

Another interesting issue is to cope with multiple representations of spatial data, i.e., allowing the same real-world object to have different geometries. Multiple representations is a common practice in spatial DBs, e.g., [40, 160]. It is also an important aspect in the context of data warehouses since spatial data may be integrated from different source systems. This integration process may either select one representation from several available (we implicitly suppose this situation in our model) or include multiple representations in a multidimensional model. Our model could be extended allowing multiple representations of spatial data. However, first several important issues must be solved. For example, the conceptual modeling of spatial data with multiple representations, for which some ideas can be taken from [13, 160]. However, if levels forming a hierarchy can have multiple representations, additional conditions may be necessary for establishing the meaningful roll-up and drill-down operations. Further, multiple-represented measures as well as a combination of multiple-represented levels and measures may also require some analysis for establishing the conditions for aggregation procedures. Some interesting insights can be obtained from [200] where the constraints for multiple-represented spatial objects related by aggregation relationships are specified.

6.2.3 Temporal and spatio-temporal data warehouses

Regarding temporal DWs, we discussed the aspect of multiple time granularities for measures in source systems and in DWs. However, complex analysis is required for management of multiple granularities between different levels forming a hierarchy and between hierarchies and measures. This is an open research area in temporal DWs. To our knowledge only [43] implicitly refer to this aspect for aggregation measures using time-varying dimensions and [184] briefly discusses issues related to multiple granularities between measures and dimension data for obtaining pre-computed aggregations of measures. On the other hand, different works related to temporal DBs propose management of multiple time granularities and data to which these time granules are attached, e.g., [16, 37, 134, 209, 212]. This research could provide solutions for managing multiple time granularities in temporal DWs.

Further, even though operations in temporal DWs have been proposed in several works (e.g., [132, 139]), none of them provide the implementations for DW data stored in an object-relational DBMS. It would be another interesting issue to implement the roll-up, drill-down, and slice-and-dice operations and evaluate them against different options for representing multidimensional schemas in object-relational databases, for example, for hierarchies (Section 4.7.2.2) or for measures (Section 4.8.3).

Finally, an important extension to our model would be to combine spatial and temporal features to propose a conceptual multidimensional model for spatio-temporal
applications supporting decision-making processes. However, combining both domains cannot be done in an automatic way; concepts proposed in this thesis should be revised while others should be included.

Currently many spatio-temporal applications refer to moving objects where the object locations in time are relevant, for example, applications related to Location-Based Services (LBSs), mobile telephone networks, transportation networks. These applications are mostly used for providing different kinds of services, therefore their data model may not require multidimensional view of data, which is used with the goal of measures aggregations while traversing hierarchies. Therefore, a first step should analyze for which spatio-temporal applications a multidimensional model is more suitable than conventional spatio-temporal models, e.g., [106, 160]. Some ideas can be taken from [156, 195, 207]. Further, the merging process of space and time may start considering different elements composing a multidimensional model proposed in this thesis. For example, temporal types for time-varying spatial dimensions and measures, aggregation of time-varying spatial measures, conditions for traversing spatial hierarchies with time-varying links and levels, and others. Many insights for this extension can be obtained from research related either to spatio-temporal DBs, e.g., [47, 73, 106, 150, 160] or spatio-temporal DWs, e.g., [18, 125, 137, 154, 155, 156, 195, 207].

6.2.4 Methodology

The DW design methodology is still an open research problem. Since requirement specifications is the most important phase in the whole process of the DW design, different approaches should be evaluated in a more formal manner. However, there is still a need to find evaluation criteria. Although, some insights can be taken from [112], their comparison of some of the methods is very empirical and applied for DW projects with different characteristics. Therefore, the future work should first consider the establishment of evaluation criteria that allow to compare different methods used for the requirements specification phase. Then, these criteria can be applied to evaluate those methods considering DWs (or data marts) with similar analysis purposes and data availability.

Another research issues is related to the synthesis of currently existing approaches used for choosing which aggregates should be precomputed. There are many scientific works that refer to this issue, for example, [5, 157]. However, they are usually highly technical and cannot be used as a general recommendation for DW developers. Currently, developers must rely on their intuition or on commercial tools, for example, Microsoft Analysis Server, which automatically selects aggregates to precompute.

Spatial and temporal DWs lack the research and practice related to their developments. Therefore, different solutions proposed by conventional data warehouses, in particular for the requirements specification phase, may be extended by spatial and temporal data. For example, for the supply-driven approach the specifications of how to derive spatially- (or temporally-) extended multidimensional schemas from the underlying spatial (or temporal) source systems could be helpful for DW designers. This extension should consider the algorithms proposed for the supply-driven method for the conventional DW design as well as research related to spatial and temporal DWs.
Appendix A

Formalization of the MultiDimER model

The following formalization is inspired from [64]. We first describe notations, assumptions, and meta variables required for defining the abstract syntax and the semantics of the MultiDimER model. Next, we give the abstract syntax of the model that allows the translation from the graphical representation to the equivalent textual representation. Then, we present some examples of using this syntax for the model in Figure A.1. Finally, after describing the auxiliary functions, we define the semantics of the MultiDimER model.

A.1 Notations and assumptions

We use \( \text{SET} \) and \( \text{TF} \) to denote the class of sets and the class of total functions, respectively. Given \( S_1, S_2, \ldots, S_n \in \text{SET} \), \( S_i \cup S_j \) indicates the disjoint union of sets, \( S_i \cup S_j \) denotes the union of sets, and \( S_1 \times S_2 \times \ldots \times S_n \) represents the Cartesian product over the sets \( S_1, S_2, \ldots, S_n \). \( \mathcal{P}(S) \) indicates powerset of a set.

The Multi dim ER model includes the basic data types \( \text{int}, \text{real}, \text{string} \). \( \text{DATA} \in \text{SET}; \{\text{int}, \text{real}, \text{string}\} \subseteq \text{DATA} \).

A.2 Meta variables

\( S_D \in \text{SchType.DECL} \) – MultiDimER schema declarations  
\( D_D \in \text{DimType.DECL} \) – dimension type declarations  
\( FR_D \in \text{FacRelType.DECL} \) – fact relationship type declarations  
\( L_D \in \text{LevType.DECL} \) – level type declarations  
\( CP_D \in \text{ChiParType.DECL} \) – child-parent type declarations  
\( H_D \in \text{Hier.DECL} \) – hierarchy declarations  
\( A_D \in \text{Att.DECL} \) – attribute declarations  
\( IC_D \in \text{IntegrityConstraint.DECL} \) – integrity constraints declarations  
\( CP_S \in \text{CPInv.SPEC} \) – the set of child-parent involvement specifications  
\( IL_S \in \text{LevInv.SPEC} \) – the set of level involvement specifications  
\( D \in \text{Dim.TYPE} \) – the set of dimension type names
A.3 Abstract syntax

\[
\begin{align*}
S_D &::= D_D; L_D; CP_D; FR_D; IC_D; \\
D_D &::= D_1; D_2 \\
&\quad| \text{Dimension type } D \text{ includes level } L \\
&\quad| \text{Dimension type } D \text{ includes } H_D \\
L_D &::= L_1; L_2 \\
&\quad| \text{Level type } L \text{ has } A_D \\
CP_D &::= CP_1; CP_2 \\
&\quad| \text{ChildParent type } CP \text{ relates } L_1 \text{ and } L_2 \\
&\quad| \text{ChildParent type } CP \text{ relates } L_1 \text{ and } L_2 \text{ with } disfac \\
&\quad| \text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } ILs \\
&\quad| \text{ChildParent type } CP \text{ relates } ILs \text{ and } L \text{ as joining level} \\
FR_D &::= FR_1; FR_2 \\
&\quad| \text{Fact relationship type } FR \text{ involves } ILs \\
&\quad| \text{Fact relationship type } FR \text{ has } A_D \text{ involves } ILs \\
IC_D &::= IC_1; IC_2 \\
&\quad| K \text{ is primary key of } L \\
&\quad| \text{Participation of } L_1 \text{ with } L_2 \text{ in } CP \text{ is } (min, max) \\
H_D &::= H_1; H_2 \\
&\quad| \text{Hierarchy } H \text{ composed of } CP_S \\
CP_S &::= CP_1, CP_2 \\
&\quad| CP \\
ILs &::= IL_1, IL_2 \\
&\quad| L \\
&\quad| L \text{ with } disfac \\
A_D &::= A_1, A_2 \\
&\quad| \text{Attribute } A \text{ of type } d \\
d &::= \text{int} | \text{real} | \text{string} \\
disfac &::= \text{real}
\end{align*}
\]
Figure A.1: Example of schema containing several hierarchies reusing hierarchy levels.
A.4. Examples using the abstract syntax

The following examples are based on the MultiDimER model of Figure A.1. Only part of the model is transformed to the textual representation.

A.4.1 Level definitions

Level type *Customer* has
- Attribute *Customer ID* of type integer,
- Attribute *Customer name* of type string,
- ...

Level type *City* has
- Attribute *City name* of type string,
- Attribute *City population* of type integer,
- ...

Level type *State* has
- Attribute *State name* of type string,
- Attribute *State population* of type integer,
- ...

A.4.2 Child-parent relationship definitions

ChildParent type *ProCat* relates *Product* and *Category*;
ChildParent type *CatDep* relates *Category* and *Department*;
ChildParent type *CusTypPro* relates *Customer* as splitting level and *Type*, *Profession*;
ChildParent type *TypSec* relates *Type* and *Sector*;
ChildParent type *ProCla* relates *Profession* and *Class*;
ChildParent type *SecClaBra* relates *Sector*, *Class* and *Branch* as joining level;
ChildParent type *BraAre* relates *Branch* and *Area*;
ChildParent type *CitConSta* relates *City* as splitting level and *County*, *State*;
ChildParent type *ConCitSta* relates *City*, *County* and *State* as joining level;
ChildParent type *StaCt* relates *State* and *Country*;
ChildParent type *EmpSec* relates *Employee* and *Section* with disfac;

A.4.3 Dimension definitions

Dimension type *Program type* includes level *Program type*;
Dimension type *Product* includes
- hierarchy *Product groups*
- composed of *ProCat*, *CatDep*;
Dimension type *Customer* includes
- hierarchy *Customer type*
composed of CusTypPro, TypSec, ProCla, SecClaBra, BraAre, 
hierarchy Customer location
composed of CusCit, CitCouSta, CouCitSta, StaCtri;
Dimension type Employee includes
hierarchy Works composed of EmpSec1, SecDw,
hierarchy Affiliated composed of EmpSec2, SecDw;

A.4.4 Fact relationship definitions

Fact relationship type Sales facts has
Attribute Sales of type real,
Attribute Quantity of type real,
involves
Time, Product, Customer, Store;
Fact relationship type Employees salaries facts has
Attribute Base salary of type real,
Attribute Working hours of type real,
Attribute Extra payment of type real,
involves
Store, Month, Employee;

A.4.5 Constraint definitions

Customer id is primary key of Customer;
Employee id is primary key of Employee;
Participation of Product with Category in ProCat is (1,1);
Participation of Employee with Section in SecEmpl is (1,n);
Participation of Section with Employee in SecEmpl is (1,n);
Participation of City with County in CitCouSta is (0,1);

A.5 Auxiliary functions

This section presents functions required for defining the semantic functions.

- Function attOfLev takes as input a level type declaration and returns the attribute names of the level:

  \[ \text{attOfLev : LevType.DECL} \rightarrow \mathcal{P}(\text{Attributes}) \]

- Function attOfFR takes as input a fact relationship type declaration and returns attribute names of the fact relationship:

  \[ \text{attOfFR : FacRelType.DECL} \rightarrow \mathcal{P}(\text{Attributes}) \]

- Function cnt takes as input a level member \( m \), a level type \( L \), and a set of instances of a child-parent relationship and returns the number of tuples in the child-parent set that the member \( m \) participates in.
A.6. SEMANTICS

In this section we define the semantics of the textual representation of the MultiDimER model.

A.6.1 Semantics of predefined data types

The semantics of predefined data type is based on a function $\mathcal{D}[\text{DATA}] \in TF$ such that $\mathcal{D}[\text{DATA}] : \text{DATA} \rightarrow \text{SET}$. We assume $\forall d \in \text{DATA} (\bot \in \mathcal{D}[\text{DATA}](d))$ where $\bot$ represents an undefined value indicating an incorrect use of a function or an error.

For basic data types (int, real, and string) we assume the usual semantics:

\[
\mathcal{D}[\text{DATA}](\text{int}) = \mathbb{Z} \cup \{\bot\}
\]
\[
\mathcal{D}[\text{DATA}](\text{real}) = \mathbb{R} \cup \{\bot\}
\]
\[
\mathcal{D}[\text{DATA}](\text{string}) = \mathbb{A}^* \cup \{\bot\}
\]

An example of the semantics of the + operator:

\[
\mathcal{D}[+] : \mathcal{D}[\text{DATA}](\text{int}) \times \mathcal{D}[\text{DATA}](\text{int}) \rightarrow \mathcal{D}[\text{DATA}](\text{int}) =
\begin{cases}
\text{if } i_1, i_2 \in \mathbb{Z} \\
\bot
\end{cases}
\]

A.6.2 Semantic domains

The MultiDimER model includes the following value domains:

$\mathcal{D}_s \cup \{\bot\}$ – the set of surrogates

$\mathcal{D}_s^L \subseteq \mathcal{D}_s$ – the set of surrogates assigned to $L \in \text{Lev}_{\text{TYPE}}$

$\mathcal{D}[\text{DATA}]$ – the set of basic domains

A.6.3 Semantic functions

In what follow we describe the signature of the semantic functions and we define the semantic functions.

\[
\mathcal{IC} : \text{Lev}_{\text{TYPE}} \rightarrow \mathcal{D}_s^L
\]
\[
\mathcal{A} : \text{Attributes} \times \text{DATA} \rightarrow \mathcal{D}[\text{DATA}]
\]
\[
\mathcal{L} : \text{Lev}_{\text{TYPE}} \times \text{Att}_{\text{DECL}} \rightarrow (\mathcal{IC}[L]) \times \mathcal{A}[\mathcal{D}]
\]
\[
\mathcal{CP} : \mathcal{P}(\text{Lev}_{\text{TYPE}}) \times \mathcal{P}(\text{Lev}_{\text{TYPE}}) \times \text{DATA} \rightarrow (\mathcal{P}(\mathcal{IC}[L]) \times \mathcal{P}(\mathcal{IC}[L])) \cup (\mathcal{P}(\mathcal{IC}[L]) \times \mathcal{D}[\text{DATA}])
\]

A.6. SEMANTICS

\[cut : D_s \times \text{Lev}_{\text{TYPE}} \times \mathcal{P}[D_s \times D_s] \rightarrow \mathcal{D}[\text{DATA}](\text{int})\]

• Predicate $\text{inSch}$ takes an argument a level type name, a child-parent type name, a dimension type name, or a fact relationship type name as well as a schema declaration. It returns $\text{true}$ if the mentioned element is declared in the schema and $\text{false}$ otherwise.

\[\text{inSch} : \text{Schema}_\text{DECL} \times \text{Schema}_\text{DECL} \rightarrow \text{PRED}\]
APPENDIX A. FORMALIZATION OF THE MULTIDIMER MODEL

The purpose of the semantic function $\mathcal{I}$C is to determine the surrogate sets of the level types that are involved in a specific relationship type, i.e., in a fact relationship type or in a child-parent type for the splitting and joining levels of generalized hierarchies. For the former, if the definition includes a distributing factor, the function $\mathcal{I}$C returns additionally the value domain of this factor, i.e., the set of real numbers.

The semantic function $C$ is a function that for a given attribute declaration, it returns the value domain of the specified data type $d$.

The following interpretation functions require additional definitions: the function $\text{dom}$ associates a set of objects $X = \{X_1, X_2, \ldots, X_n\}$ with the set of value domains $D$ ($\text{dom} : X \rightarrow D$). The instances (tuples) are represented as a set of functions. The domain of each function $t$ is a set of objects $X$. Each function $t$ associates the specific object $X_i$ with a value from the value domain $D_i$. We denote it as $t[X_i] : X_i \rightarrow \text{dom}(X_i)$.

The purpose of the semantic function $C\mathcal{P}$ is to determine a set of relationships existing between members of the level types specified in the declaration. Members are identified through their surrogates with the value domain defined by $\mathcal{I}$C.
real numbers. If the declaration of the child-parent type refers to the splitting level (respectively, joining) of generalized hierarchies, the set of returned tuples contains surrogate values of members belonging to different parent (respectively, child) level types.

Since each child-parent type defines a unique set of tuples, if the function $CP$ applies to a composition of child-relationship type definitions, it returns the disjoint union of the functions applied to each of the component.

$$CP[CP_{D_1}; CP_{D_2}] = CP[CP_{D_1}] \cup CP[CP_{D_2}]$$

The semantic function $FTZ$ identifies the tuples specified through the definition of the fact relationship type $FR$. It determines a set of relationships existing among members from different level types specified by $IL_S$. Members are identified through their surrogates with the value domain defined by $IL$. If the declaration of fact relationship type $FR$ contains attributes, the domain of the function $t$ includes additionally the set of attribute names. The value domains of these attributes are determined by the semantics of the attribute declarations.

As for the level and the child-parent types, a fact relationship type defines a unique set of tuples that are stored separately in the database.

$$FR[FR_{P_1}; FR_{P_2}] = FR[FR_{P_1}] \cup FR[FR_{P_2}]$$

The semantic function $C$ ensures that the designed database is valid. It guarantees that the constructs mentioned in the constraint declaration are in the schema definition. It also defines the predicates that the objects involved in the constraint have to satisfy. The result of the application of a $C$ function to a set of constraints is a set of predicates that the database must satisfy.

In the textual representation, all constraints are separate constructs, therefore, for each of them we first should check whether different elements mentioned in the constraint, e.g., level types, child-parent types, exist at all.

$$C[IC_{D_1}; IC_{D_2}] = C[IC_{D_1}] \land C[IC_{D_2}]$$
APPENDIX A. FORMALIZATION OF THE MULTIDIMER MODEL

• Primary key constraint.
This constraint ensures that the values of the key attributes are unique for the
level set.
\[ C[\text{K is primary key of } L] = \text{inSch}(L, S_D) \land \]
\[ K \in \text{attr}(\text{Lev}(\text{Level type } L \text{ has } A_D)) \land \]
\[ \forall t_i, t_j \in C[\text{Level type } L \text{ has } A_D] \ (t_i[K] = t_j[K] \rightarrow t_i[s] = t_j[s]) \]

• A child member can be related to minimum \( \text{min} \) and maximum \( \text{max} \) parent
members.
First, we take into account different kinds of child-parent type declarations and
refer to the level that plays the rôle of child. Then, we apply the function \( \text{cnt} \) to
determine the number of tuples in the set of members of the child-parent type
that member \( m \) of the child level type \( L \) participates in. Finally, this number is
compared with the given minimum and maximum values.
\[ C[\text{Participation of } L \text{ with } L' \text{ in } CP \text{ is } (\text{min}, \text{max})] = \]
\[ \text{inSch}(L, S_D) \land \text{inSch}(L', S_D) \land \text{inSch}(CP, S_D) \land \]
\[ (CP_D = \text{ChildParent type } CP \text{ relates } L \text{ and } L') \lor \]
\[ (CP_D = \text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } \]
\[ IL_S \land L' \in IL_S) \lor \]
\[ (CP_D = \text{ChildParent type } CP \text{ relates } IL_S \]
\[ \text{and } L \text{ as joining level } \land L \in IL_S)) \land \]
\[ \forall m \in D_{IL}^L (\text{min} \leq \text{cnt}(m, L, CP[CP_D]) \leq \text{max}) \]

• A parent member can be related to minimum \( \text{min} \) and maximum \( \text{max} \) child
members.
In order to determine the predicates for this constraint, we apply similar steps as
explained for the previous constraint with the exception of considering now the
level that plays the rôle of parent instead of child.
\[ C[\text{Participation of } L \text{ with } L' \text{ in } CP \text{ is } (\text{min}, \text{max})] = \]
\[ \text{inSch}(L, S_D) \land \text{inSch}(L', S_D) \land \text{inSch}(CP, S_D) \land \]
\[ (CP_D = \text{ChildParent type } CP \text{ relates } L' \text{ and } L) \lor \]
\[ (CP_D = \text{ChildParent type } CP \text{ relates } L' \text{ as splitting level and } \]
\[ IL_S \land L \in IL_S) \lor (CP_D = \text{ChildParent type } CP \text{ relates } IL_S \]
\[ \text{and } L \text{ as joining level } \land L \in IL_S)) \land \]
\[ \forall m \in D_{IL}^L (\text{min} \leq \text{cnt}(m, L, CP[CP_D]) \leq \text{max}) \]

• A member of a splitting level is related to members of at most one parent level.
First, we consider the child-parent type declaration related to the splitting level.
Then, for every pair of surrogates belonging to the set of members of this child-
parent type we check the following: if the member of the child level type is related
to a member of some parent level type, there is not another parent level type with
a member that is related to the same child member.
\[ C[\text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } IL_S] = \]
\[ \text{inSch}(L, S_D) \land \forall m \in D_{IL}^L \forall L_i \in IL_S \forall m_i \in D_{IL}^L \]
\[ (\text{inSch}(L_i, S_D) \land (m, m_i) \in CP[\text{ChildParent type } \]
\[ CP \text{ relates } L \text{ as splitting level and } IL_S]) \rightarrow \]
\[-(\exists L_j \in IL_S \exists m_j \in D_{IL}^L (L_i \neq L_j \land \]}
A.6. SEMANTICS

$$(m, m_s) \in \mathcal{CP} [\text{ChildParent type CP relates } L \\
asplitting level and } IL_S]$$

The last of the semantic function $S$ specifies that the MultiDimER schema is composed of definitions of level types, child-parent types, fact relationship types, and integrity constraints. It defines each component of the underlying database and predicates that ensure validity and consistency of the database.

$$S[S_D] = S[L_D; CP_D; FR_D; IC_D]$$
$$S[L_D; CP_D; FR_D; IC_D] = \mathcal{L}[L_D] \cup \mathcal{CP}[CP_D] \cup \mathcal{FR}[FR_D] \cup \mathcal{IC}[IC_D]$$

Notice that in the above formalization dimensions and hierarchies do not have semantic interpretations. However, dimensions are required for the drill-across operation that allows to expand analysis comparing measures from different fact relationships. Hierarchies are needed for defining meaningful aggregations for the roll-up and drill-down operations. Such operations are beyond the scope of this paper.
Appendix B

Formalization of the spatially-extended MultiDimER model

The following formalization extends the formalization presented in Appendix A by the inclusion of spatial elements. We will use the same notations and assumptions as the ones presented in Appendix A. Further, since meta variables, syntax, and semantics of the non-spatial part of the model are the same as presented in Appendix A, we will repeat those that facilitate the understanding of the model.

B.1 Meta variables for the spatial extension

The following meta variables are introduced additionally to the meta variables presented in Appendix A.

$G_s \in Geo\_SPEC$ – the set of specification for spatial support

$G_e \in GEO = \{point, line, surface\}$ – the set of primitive geometry types

B.2 Abstract syntax

We present abstract syntax in their totality to give a general view of the proposed model. Notice that since spatial support is considered for levels and attributes\(^1\), only two new declarations are included.

\[
\begin{align*}
S_D & ::= D_D; L_D; CP_D; FR_D; IC_D; \\
D_D & ::= D_D_1; D_D_2 \\
L_D & ::= L_D_1; L_D_2 \\
\end{align*}
\]

\(^1\)Recall that measures are presented as attributes.
B.3. Examples using the abstract syntax

The following example (Figure B.1) extends the example from Figure 3.2 by including an additional fact relationship and dimensions for analyzing repairing costs that different administrative units must cover. Only a part of the model is given in the textual representation.

B.3.1 Level definitions

Level type Road coating has

Attribute Coating name of type string,
Attribute Coating type of type string,

Level type County with geometry surface has

Attribute County name of type string,
Attribute County population of type integer,
Attribute County area of type real,
Figure B.1: Example of a model including spatial and non-spatial elements.

Level type State with geometry set surface has
  Attribute State name of type string,
  Attribute State population of type integer,
  Attribute State area of type real,
  Attribute State major activity of type string,
  Attribute Capital of type string with geometry point;

Level type Highway segment with geometry line has
  Attribute Segment number of type string,
  Attribute Road condition of type string,
  Attribute Speed limit of type real;

Level type Administr unit has
  Attribute Unit name of type string,
  Attribute Description of type string,
  Attribute Responsible of type string,
  Attribute Address of type string with geometry point;
B.3. EXAMPLES USING THE ABSTRACT SYNTAX

B.3.2 Child-parent relationship definitions

ChildParent type HSegHSec relates Highway segment and Highway section;
ChildParent type HSecH relates Highway section and Highway;
ChildParent type CouSta relates County and State;
ChildParent type AdmCouSta relates Administr unit as splitting level and County, State;
ChildParent type CouAdmSta relates Administr unit, County and State as joining level;

B.3.3 Dimension definitions

Dimension type Road coating includes level Road coating;
Dimension type Highway segment includes hierarchy Highway structure composed of HSegHSec, HSecH;
Dimension type County includes hierarchy Geo location composed of CouSta;
Dimension type Administr unit includes hierarchy Organiz division composed of AdmCouSta, CouAdmSta;

B.3.4 Fact relationship definitions

Fact relationship type Highway maintenance has
Attribute Length of type real,
Attribute Common area of type real with geometry line,
Attribute No. cars of type integer,
Attribute Repair cost of type real;

involves
Highway segment, Road coating, County, Time;

Fact relationship type Highway repairs has
Attribute Cost of type real,
Attribute Quantity of type real;

involves
Time, Administr unit, Material;

B.3.5 Constraint definitions

Segment number is primary key of Highway segment;
Material name is primary key of Material;
Participation of Highway segment with Highway section in HSegHSec is (1,1);
Participation of Highway section with Highway segment
APPENDIX B. FORMALIZATION OF THE SPATIAL MULTIDIMER MODEL

in HSegHSec is \((1,n)\);
Participation of State with Administr. unit in CouAdmSta is \((0,n)\);

B.4 Semantics

In this section we define the semantics of the textual representation of the MultiDimER model. First, we define the model of space. Next, we present the full semantics of the spatially-extended MultiDimER model excepted the semantics of the predefined data types since they are the same as those presented in Appendix A.

B.4.1 The space model

Euclidean space is used (or implicitly assumed) as a basis for modeling spatial objects. Essentially, this means that a point in the plane is given by a pair of real numbers. Unfortunately, real numbers cannot be represented in computers, only their finite approximations. This leads to problems in geometric computations, i.e., errors in query processing [69]. Therefore, some authors, for example, [45, 71] have suggested introducing a discrete basis for spatial object modeling as well as implementation. Our model uses the finite representation based on a concept of a realm [71, 72]. A realm is in general a finite, user-defined structure that is used as a basis for one or more system data types. Realms are somewhat similar to enumeration types in programming languages. A realm used as basis for spatial data types is essentially a finite set of points and non-intersecting line segments. All points, lines, and surfaces required for creating spatial objects can be defined in terms of points and line segments present in the realm. In fact, in a database spatial objects are never created directly but are formed by selecting some realm objects.

Next, we present the formal definition of a realm [71]:

Let \(N = \{0, \ldots, n-1\} \subseteq \mathbb{N}\). An \(N\)-point is a pair \((x, y) \in N \times N\). An \(N\)-segment is a pair of distinct \(N\)-points \((p, q)\). \(P_N\) denotes a set of all \(N\)-points and \(S_N\) denotes a set of all \(N\)-segments. Given \(N\), a realm over \(N\), or \(N\)-realm, is a set \(R = P \cup S\) such that:

- \(P \subseteq P_N, S \subseteq S_N\)
- \(\forall s \in S : s = (p, q) \Rightarrow p \in P \land q \in P\)
- \(\forall p \in P \forall s \in S : \neg(p \text{ in } s)\), where the \(\text{in}\) operation tests if an \(N\)-points lies on an \(N\)-segment
- \(\forall s, t \in S : \neg(s = t) \land \neg(s \text{ intersect } t) \land \neg(s \text{ overlap } t)\), where two \(N\)-segments overlap if they share a (partial) \(N\)-segment and intersect if they have exactly one common point different from end points.

In other words, a realm conceptually describes the complete underlying geometry of a particular spatial object in two dimensions: (i) it is a finite set of points and line segments over a discrete grid, (ii) each end point of a realm segment is also a point of the realm, (iii) no realm point lies within a realm segment, and (iv) no two realm
segments intersect except at their end points [70]. Figure B.2 illustrates a realm [69]. Notice that different lines or surfaces can be defined using this realm.

The ROSE (RObust Spatial Extensions) algebra offers three data types called points, lines, and regions whose values are realm-based (i.e., composed from elements of a realm) together with a comprehensive set of operations [70]. To define the domains of these data types, intermediate notations of an $R$-point, $R$-block, and $R$-face are required [69].

For a given realm $R$, an $R$-point is a point of $R$. An $R$-block is a connected set of line segments of $R$. An $R$-face is essentially a polygon with holes that can be defined over realm segments. Then, a domain of a type points is a set of $R$-points, a domain of a type lines is a set of disjoint $R$-blocks, and a domain of a type regions is a set of edge-disjoint $R$-faces (where edge-disjoint means that two faces may have a common vertex, but no common edge) [70].

Since realms and realm-based spatial objects are defined over a finite discrete space $N \times N$ with $N = \{0, \ldots, m-1\} \subseteq \mathbb{N}$, points, end points of lines, vertices of lines forming regions, etc., have integer coordinates instead of arbitrary floating-point coordinates [70]. Further, [72] define algebraic operations for the spatial data types to construct only geometric objects that are realm-based as well. So the spatial algebra is closed with respect to a given realm.

Adapting [71, 72] to our notations, we assume the existence of three spatial data types: $GEO = \{point, line, surface\}$. For each of these types we define value domains that represent corresponding sets as explained above:

- $D_{points} = \{R$-points\} - the point domain
- $D_{lines} = \{R$-block\} - the line domain
- $D_{surfaces} = \{R$-face\} - the surface domain

Therefore, based on the previous explanations, we can define the semantics interpretation of spatial data types as follows:

- $D(GEO)(point) = N \times N \cup \{\bot\}$
- $D(GEO)(line) = P(N \times N) \cup \{\bot\}$
- $D(GEO)(surface) = P(N \times N) \cup \{\bot\}$
B.4.2 Semantic domains

The MultiDimER model includes the following value domains:

- $D_s \cup \{\bot\}$ – the set of surrogates
- $D_s^L \subseteq D_s$ – the set of surrogates assigned to $L \in \text{Lev\_TYPE}$
- $\mathcal{D}[\text{DATA}]$ – the set of basic domains
- $\mathcal{D}[\text{GEO}]$ – the set of basic spatial domains

B.4.3 Semantic functions

In the following we use auxiliary functions defined in Appendix A. First, we describe the signature of the semantic functions and then, we define these semantic functions.

- $\mathcal{IC} : \text{Lev\_TYPE} \times \mathcal{D}_\text{spec} \rightarrow \mathcal{D}_\text{points} \cup \mathcal{D}_\text{lines} \cup \mathcal{D}_\text{surfaces}$
- $\mathcal{A} : \text{Attributes} \times \mathcal{D}_\text{DATA} \times \mathcal{D}_\text{SPEC} \rightarrow \mathcal{D}[\text{DATA}] \cup (\mathcal{G}[\mathcal{G}_s] \rightarrow \mathcal{D}[\text{DATA}])$
- $\mathcal{C} : \text{Lev\_TYPE} \times \mathcal{D}_\text{SPEC} \times \text{Att\_DECL} \rightarrow \mathcal{I}_\mathcal{C}[\mathcal{L}] \times \mathcal{A}[\mathcal{A}_d] \cup (\mathcal{G}[\mathcal{G}_s] \rightarrow \mathcal{I}_\mathcal{C}[\mathcal{L}] \times \mathcal{A}[\mathcal{A}_d])$
- $\mathcal{C}_\mathcal{P} : \mathcal{P}(\text{Lev\_TYPE}) \times \mathcal{P}(\text{Lev\_TYPE}) \times \mathcal{D}_\text{DATA} \rightarrow (\mathcal{P}(\mathcal{I}_\mathcal{C}[\mathcal{L}]) \times \mathcal{P}(\mathcal{I}_\mathcal{C}[\mathcal{L}])) \cup (\mathcal{P}(\mathcal{I}_\mathcal{C}[\mathcal{L}]) \times \mathcal{P}(\mathcal{I}_\mathcal{C}[\mathcal{L}]) \times \mathcal{D}[\text{DATA}])$
- $\mathcal{F}_\mathcal{R} : \text{Lev\_inv\_SPEC} \times \text{Att\_DECL} \rightarrow \mathcal{I}_\mathcal{C}[\mathcal{I}_\mathcal{L}_s] \cup (\mathcal{I}_\mathcal{C}[\mathcal{I}_\mathcal{L}_s] \times \mathcal{A}[\mathcal{A}_d])$
- $\mathcal{C} : \mathcal{I}_D \rightarrow \mathcal{PRED}$
- $\mathcal{S} : \mathcal{S}_D \rightarrow \mathcal{S}[\mathcal{S}_D]$

The purpose of the semantic function $\mathcal{IC}$ is to determine the surrogate sets of the level types that are involved in a specific relationship type, i.e., in a fact relationship type or in a child-parent type for the splitting and joining levels of generalized hierarchies. For the former, if the definition includes a distributing factor, the function $\mathcal{IC}$ returns additionally the value domain of this factor.

- $\mathcal{I}_\mathcal{C}[\mathcal{I}_\mathcal{L}_s, \mathcal{I}_\mathcal{L}_s] = \mathcal{I}_\mathcal{C}[\mathcal{I}_\mathcal{L}_s] \times \mathcal{I}_\mathcal{C}[\mathcal{I}_\mathcal{L}_s]$
- $\mathcal{I}_\mathcal{C}[\mathcal{L}] = \begin{cases} D_s^L & \text{if } L \in \text{Lev\_TYPE} \\ \bot & \text{otherwise} \end{cases}$
- $\mathcal{I}_\mathcal{C}[\mathcal{L} \text{ with } \text{disfac} = \mathcal{C} \{ (D_s^L, \mathcal{D}[\text{DATA}](\text{disfac})) \text{ if } L \in \text{Lev\_TYPE} \land \text{disfac} \in \mathbb{R} \text{ otherwise} \}$

The function $\mathcal{G}$ returns the value of spatial domain according to a particular declaration of geometry type.

- $\mathcal{G} : \mathcal{G}_s, \mathcal{G}_s = \mathcal{G}[\mathcal{G}_s_1] \times \mathcal{G}[\mathcal{G}_s_2]$
- $\mathcal{G}[\text{gele}] = \mathcal{D}[\text{GEO}](\text{gеле})$
- $\mathcal{G}[\text{set } \text{gele}] = \mathcal{P}(\mathcal{D}[\text{GEO}](\text{gеле}))$

The semantic function $\mathcal{A}$ is a function that for a given attribute declaration, it returns the value domain of the specified data type $d$. If the attribute of type $d$ includes spatial support, this indicates that the value of this attribute is represented in the space. Therefore, the value domain of this attribute must be defined as a function from space domain to a value domain.

- $\mathcal{A}[\mathcal{A}_d_1, \mathcal{A}_d_2] = \mathcal{A}[\mathcal{A}_d_1] \times \mathcal{A}[\mathcal{A}_d_2]$
B.4. SEMANTICS

\[ A[\text{Attribute } A \text{ of type } d] = \begin{cases} \mathcal{D}[\text{DATA}](d) & \text{if } d \in \text{DATA} \\ \bot & \text{otherwise} \end{cases} \]

\[ A[\text{Attribute } A \text{ of type } d \text{ with geometry } G_s] = \begin{cases} \mathcal{G}[G_s] \rightarrow \mathcal{D}[\text{DATA}](d) & \text{if } d \in \text{DATA} \land G_s \in \text{GeoSPEC} \\ \bot & \text{otherwise} \end{cases} \]

The following interpretation functions require additional definitions: the function \( \text{dom} \) associates a set of objects \( X = \{X_1, X_2, \ldots, X_n\} \) with the set of value domains \( \mathcal{D} (\text{dom} : X \rightarrow D) \). The instances (tuples) are represented as a set of functions. The domain of each function \( t \) is a set of objects \( X \). Each function \( t \) associates the specific object \( X_i \) with a value from the value domain \( D_i \). We denote it as \( t[X_i] : X_i \rightarrow \text{dom}(X_i) \).

The semantic function \( \mathcal{L} \) identifies the set of functions \( t \) (tuples) that are specified through the definition of level types. The domain of each function \( t \) is the set of attribute names belonging to the level type \( L \) and the surrogate attribute \( s \). The value domain of the surrogate attribute \( s \) is the set \( D_g \) of surrogate values assigned to the level \( L \) while the value domain of the attributes of the level \( L \) is determined by the semantics of the attribute declarations.

If the level has defined spatial support, this means that the level members are represented in a space. Similar as was done for spatial support of attributes, the value domain of the surrogate attribute is defined as a function from space domain to a set of surrogate values.

The semantic function \( \mathcal{L} \) applied to a composition of level type definitions returns the disjoint union of the functions applied to each of the component. This is since each level type defines a unique set of tuples, which is stored separately in the database.

\[ \mathcal{L}[L_{D_1}; L_{D_2}] = \mathcal{L}[L_{D_1}] \cup \mathcal{L}[L_{D_2}] \]

\[ \mathcal{L}[\text{Level type } L \text{ has } A_D] = \{ t | t \in \text{TF} \land \text{dom}(t) = \{s, \text{attOfLev}(L_D)\} \wedge t[s] \in D_g \land \forall A_i \in \text{attOfLev}(L_D) (t[A_i] \in A[\text{Attribute } A \text{ of type } d] \vee t[A_i] \in A[\text{Attribute } A \text{ of type } d \text{ with geometry } G_s]) \} \]

\[ \mathcal{L}[\text{Level type } L \text{ with geometry } G_s \text{ has } A_D] = \{ t | t \in \text{TF} \land \text{dom}(t) = \{s, \text{attOfLev}(L_D)\} \wedge t[s] \in \mathcal{G}[G_s] \rightarrow D_g \land \forall A_i \in \text{attOfLev}(L_D) (t[A_i] \in A[\text{Attribute } A \text{ of type } d] \vee t[A_i] \in A[\text{Attribute } A \text{ of type } d \text{ with geometry } G_s]) \} \]

The purpose of the semantic function \( \mathcal{CP} \) is to determine a set of relationships existing between members of the level types specified in the declaration. Members are identified through their surrogates with the value domain defined by \( \mathcal{L} \).

The function \( \mathcal{CP} \) is applicable to all types of child-parent type definitions. If the definition of this relationship includes a distributing factor, the domain of the function \( t \) includes additionally an attribute representing it (\( p \)); its value domain is a set of real numbers. If the declaration of the child-parent type refers to the splitting level (respectively, joining) of generalized hierarchies, the set of returned tuples contains surrogate values of members belonging to different parent (respectively, child) level types.

Since each child-parent type defines a unique set of tuples, if the function \( \mathcal{CP} \) applies to a composition of child-relationship type definitions, it returns the disjoint union of
the functions applied to each of the component.

The semantic function \( \mathcal{F}_R \) identifies the tuples specified through the definition of the fact relationship type \( FR \). It determines a set of relationships existing among members from different level types specified by \( IL_S \). Members are identified through their surrogates with the value domain defined by \( IL \). If the declaration of fact relationship type \( FR \) contains attributes, the domain of the function \( t \) includes additionally the set of attribute names. The value domains of these attributes are determined by the semantics of the attribute declarations.

As for the level and the child-parent types, a fact relationship type defines a unique set of tuples that are stored separately in the database.

The semantic function \( \mathcal{C} \) ensures that the designed database is valid. It guarantees that the constructs mentioned in the constraint declaration are in the schema definition. It also defines the predicates that the objects involved in the constraint have to satisfy. The result of the application of a \( \mathcal{C} \) function to a set of constraints is a set of predicates that the database must satisfy.

In the textual representation, all constraints are separate constructs, therefore, for each of them we first should check whether different elements mentioned in the constraint, e.g., level types, child-parent types, exist at all.

- Primary key constraint.
  This constraint ensures that the values of the key attributes are unique for the level set.

\[ \mathcal{C}[K \text{ is primary key of } L] = inSch(L, SD) \]
B.4. SEMANTICS

\[ K \in attOfLev(\text{Level type } L \text{ has } A_D) \land \\
\forall t_i, t_j \in C[\text{Level type } L \text{ has } A_D] \ (t_i[K] = t_j[K] \rightarrow t_i[s] = t_j[s]) \]

- A child member can be related to minimum \( \text{min} \) and maximum \( \text{max} \) parent members.

First, we take into account different kinds of child-parent type declarations and refer to the level that plays the role of child. Then, we apply the function \( \text{cnt} \) to determine the number of tuples in the set of members of the child-parent type that member \( m \) of the child level type \( L \) participates in. Finally, this number is compared with the given minimum and maximum values.

\[ C[\text{Participation of } L \text{ with } L' \text{ in } \text{CP is } (\text{min}, \text{max})] = \\
inSch(L, S_D) \land inSch(L', S_D) \land inSch(CP, S_D) \land \\
(CP_D = \text{ChildParent type } CP \text{ relates } L \text{ and } L') \lor \\
(CP_D = \text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } \\
IL_s \land L' \in IL_s) \lor \\
(CP_D = \text{ChildParent type } CP \text{ relates } IL_s \text{ and } L' \text{ as joining level } \land L \in IL_s) \land \\
\forall m \in D^L_s (\text{min} \leq \text{cnt}(m, L, CP[CP_D]) \leq \text{max}) \]

- A parent member can be related to minimum \( \text{min} \) and maximum \( \text{max} \) child members.

In order to determine the predicates for this constraint, we apply similar steps as explained for the previous constraint with the exception of considering now the level that plays the role of parent instead of child.

\[ C[\text{Participation of } L \text{ with } L' \text{ in } \text{CP is } (\text{min}, \text{max})] = \\
inSch(L, S_D) \land inSch(L', S_D) \land inSch(CP, S_D) \land \\
(CP_D = \text{ChildParent type } CP \text{ relates } L' \text{ and } L) \lor \\
(CP_D = \text{ChildParent type } CP \text{ relates } L' \text{ as splitting level and } \\
IL_s \land L \in IL_s) \lor (CP_D = \text{ChildParent type } CP \text{ relates } IL_s \text{ and } L' \text{ as joining level } \land L \in IL_s) \land \\
\forall m \in D^L_s (\text{min} \leq \text{cnt}(m, L, CP[CP_D]) \leq \text{max}) \]

- A member of a splitting level is related to members of at most one parent level.

First, we consider the child-parent type declaration related to the splitting level. Then, for every pair of surrogates belonging to the set of members of this child-parent type we check the following: if the member of the child level type is related to a member of some parent level type, there is not another parent level type with a member that is related to the same child member.

\[ C[\text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } IL_s] = \\
inSch(L, S_D) \land \forall m \in D^L_s \forall L_i \in IL_s \forall m_i \in D^L_s \\
(inSch(L_i, S_D) \land (m, m_i) \in CP[\text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } IL_s]) \rightarrow \\
(\exists L_i \in IL_s \exists m_i \in D^L_s (L_i \neq L_j \land \\
(m, m_i) \in CP[\text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } IL_s])) \]

The last of the semantic function \( S \) specifies that the MultiDimER schema is composed of definitions of level types, child-parent types, fact relationship types, and in-
tegrity constraints. It defines each component of the underlying database and predicates that ensure validity and consistency of the database.

\[
S[S_D] = S[L_D; CP_D; FR_D; IC_D] \\
S[L_D; CP_D; FR_D; IC_D] = L[L_D] \uplus CP[CP_D] \uplus FR[FR_D] \uplus IC[IC_D]
\]
Appendix C

Formalization of the temporally-extended MultiDimER model

In this appendix, similar to Appendix B, we extend the formalization presented in Appendix A by the inclusion of temporal elements. We use the same notations and assumptions as the ones presented in Appendix A. Since meta-variables, syntax, and semantics of the non-temporal part of the model are the same as presented in Appendix A, we will repeat those that facilitate the understanding of the model.

C.1 Meta variables for the temporal extension

$T_S \in T\_{SPEC}$ – the set of specification for temporal support
$t_{ele} \in \{LS, VT, TT, DWLT\}$ – the set of temporal types
$tt \in \{instant, temporal element\}$ – the set of data types for timestamps
$gr \in \{sec, min, hour, day, week, month, year\}$ – the set of granules

C.2 Abstract syntax

\[
\begin{align*}
S_D &::= D_D; L_D; CP_D; FR_D; IC_D; \\
D_D &::= D_{D_1}; D_{D_2} \\
&\mid \text{Dimension type } D \text{ includes level } L \\
&\mid \text{Dimension type } D \text{ includes hierarchy } H_D \\
L_D &::= L_{D_1}; L_{D_2} \\
&\mid \text{Level type } L \text{ has } A_D \\
&\mid \text{Level type } L \text{ with } T_S \text{ has } A_D \\
CP_D &::= CP_{D_1}; CP_{D_2} \\
&\mid \text{ChildParent type } CP \text{ relates } L_1 \text{ and } L_2 \\
&\mid \text{ChildParent type } CP \text{ relates } L_1 \text{ and } L_2 \text{ with } disf_{ac} \\
&\mid \text{ChildParent type } CP \text{ relates } L_1 \text{ and } L_2 \text{ during } T_S \text{ with } disf_{ac}
\end{align*}
\]

231
APPENDIX C. FORMALIZATION OF THE TEMPORAL MULTIDIMENSIONAL MODEL

ChildParent type \( CP \) relates \( L \) as splitting level and \( IL_s \)
ChildParent type \( CP \) relates \( IL_s \) and \( L \) as joining level

\[
FR_D ::= \frac{FR_D_1; FR_D_2}{\text{Fact relationship type } \forall \text{ involves } IL_s}
\]

\[
IC_D ::= \frac{IC_D_1; IC_D_2}{K \text{ is primary key of } L}
\]

| \text{Snapshot participation of } L_1 \text{ with } L_2 \text{ in } CP \text{ is } (\text{min}, \text{max}) |
| \text{Lifespan participation of } L_1 \text{ with } L_2 \text{ in } CP \text{ is } (\text{min}, \text{max}) |

\[
H_D ::= \frac{H_D_1, H_D_2}{\text{Hierarchy } H \text{ composed of } CP_S}
\]

\[
CP_S ::= \frac{CP_{S_1}, CP_{S_2}}{CP}
\]

\[
IL_S ::= \frac{IL_{S_1}, IL_{S_2}}{L}
\]

\[
\text{L with disfac}
\]

\[
\text{L during } T_s
\]

\[
\text{L during } T_s \text{ with disfac}
\]

\[
A_D ::= \frac{A_{D_1}, A_{D_2}}{\text{Attribute } A \text{ of type } d}
\]

\[
\text{Attribute } A \text{ of type } d \text{ with } T_s
\]

\[
T_S ::= \frac{T_{S_1}, T_{S_2}}{(t_{\text{ele}}, t_S, gr)}
\]

\[
t_{\text{ele}} ::= \text{LS | VT | TT | DWLT}
\]

\[
t_S ::= \text{instant | temporal element}
\]

\[
gr ::= \text{sec | min | hour | day | month | year}
\]

\[
d ::= \text{int | real | string}
\]

\[
\text{disfac} ::= \text{real}
\]

C.3 Examples using the abstract syntax

The example in Figure C.1 extends the one from Figure 4.2 by including an additional fact relationship and dimensions. Only part of the model is given in the textual representation.

C.3.1 Level definitions

Level type \( \text{Customer} \) has

\begin{align*}
\text{Attribute } \text{Customer ID} & \text{ of type integer}, \\
\text{Attribute } \text{Customer name} & \text{ of type string}, \\
\end{align*}

\ldots

Level type \( \text{Product} \) with \( (LS, \text{temporal element, month}) \) has

\begin{align*}
\text{Attribute } \text{Product number} & \text{ of type integer}, \\
\text{Attribute } \text{Product name} & \text{ of type string}, \\
\text{Attribute } \text{Description} & \text{ of type string},
\end{align*}
Figure C.1: Example of a model including temporal and non-temporal elements.
APPENDIX C. FORMALIZATION OF THE TEMPORAL MULTIDIMER MODEL

Attribute Size of type real with (VT, temporal element, month),
Attribute Distributor name of type string with (VT, temporal element, month);

Level type City with (LS, temporal element, year) has
Attribute City name of type string,
Attribute City population of type integer with (VT, temporal element, year),
Attribute City area of type integer with (VT, temporal element, year),
...

Level type County with (LS, temporal element, year) has
Attribute County name of type string,
Attribute County population of type integer with (VT, temporal element, year),
Attribute County area of type integer with (VT, temporal element, year),
...

Level type State with (LS, temporal element, year) has
Attribute State name of type string with (VT, temporal element, year),
Attribute State population of type integer with (VT, temporal element, year),
...

Level type Employee with (LS, temporal element, month), (TT, temporal element, month) has
Attribute Employee id of type string,
Attribute First name of type string,
...
Attribute Position of type string with (VT, temporal element, month), (TT, temporal element, month),
Attribute Title of type string with (VT, temporal element, month), (TT, temporal element, month);

C.3.2 Child-parent relationship definitions

ChildParent type ProCat relates Product and Category during (VT, temporal element, year);
ChildParent type CusCit relates Customer and City during (VT, temporal element, year);
ChildParent type CitCouSta relates City as splitting level and County during (VT, temporal element, year), State during (VT, temporal element, year);
ChildParent type CouCitSta relates City during (VT, temporal element, year), County during (VT, temporal element, year),
and State as joining level;
ChildParent type StaCitr relates State and Country during
(VT, temporal element, year);
ChildParent type EmpSec1 relates Employee and Section during
(VT, temporal element, year);
ChildParent type EmpSec2 relates Employee and Section during
(VT, temporal element, year) with disfac;

C.3.3 Dimension definitions

Dimension type Product includes
  hierarchy Product groups
  composed of ProCat;
Dimension type Customer includes
  hierarchy Customer location
  composed of CusCit, CitCusSta, ConCitSta, StaCitr;
Dimension type Employee includes
  hierarchy Works composed of EmpSec1,
  hierarchy Affiliated composed of EmpSec2;

C.3.4 Fact relationship definitions

Fact relationship type Sales Facts has
  Attribute Sales of type real with (DWLT, instant, month),
  Attribute Quantity of type real with (DWLT, instant, month)
  involves
  Product, Customer, Store;
Fact relationship type Employee Facts has
  Attribute Base salary of type real
  with (VT, temporal element, month),
  Attribute Working hours of type real
  with (VT, temporal element, month),
  Attribute Extra payment of type real
  with (VT, temporal element, month)
  involves
  Store, Employee;

C.3.5 Constraint definitions

Store number is primary key of Store;
Snapshot participation of Product with Category
  in ProCat is (1,1);
Lifespan participation of Product with Category
  in ProCat is (1,n);
Snapshot participation of Employee with Section.
Appendix C. Formalization of the Temporal Multidimer Model

In EmpSec1 is \((t,n)\);
Lifespan participation of Employee with Section
in EmpSec1 is \((t,n)\);

C.4 Auxiliary functions

This section presents additional functions required for defining the semantic constraints for the temporally-extended MultiDimER model. The specifications of other auxiliary functions used for a non-temporal version of the model are included in Appendix A.

- The function \(\text{tempSpec} \) takes as argument a child-parent declaration and returns the specification of required temporal support if the declaration is temporal and empty set otherwise.

\[ \text{tempSpec} : \text{ChiPartType.DECL} \rightarrow \text{T.SPEC} \]

- The function \(\text{tempLink} \) takes as argument a set of level involvement specifications and returns the level name if the declaration is temporal and empty set otherwise.

\[ \text{tempLink} : \text{LevInv.SPEC} \rightarrow \text{Lev.TYPE} \]

C.5 Semantics

In this section we define the semantics of the textual representation of the MultiDimER model. First, we define the semantics of the time model [64]. Next, we present the full semantics of the temporally-extended MultiDimER model except the semantics of the predefined data types since they were already specified in Appendix A.

C.5.1 The time model

We assume a time domain that is discrete and bounded [53, 64, 93]. Time domain is ordered, finite sets of elements isomorphic to finite subset of the integer numbers. The non-decomposable elements of the time domain are called chronons. Depending on application requirements consecutive chronons can be grouped into a larger unit called a granule, such as a second, a minute, or a day. Granularity represents the size of the granule, i.e., it is the time unit used for specifying the duration of the granule. We denote a granule as \(g\). The granule \(g_{\text{now}}\) denotes the granule representing current time.

Following Gregersen and Jensen [64], we include a domain for each combination of the temporal types \(t_{\text{ele}} \in \{LS, VT, TT, DWLT\}\) and granularities \(gr\). These domains are denoted \(D^g_{\text{ele}} = \{g_1^{\text{ele}}, g_2^{\text{ele}}, \ldots, g_n^{\text{ele}}\}\), e.g., \(D_{\text{month}} = \{\text{Jan, Feb, Mar, ..., Dec}\}\). The domain of each temporal type is the union of domains represented by different granularities: \(D_{\text{ele}} = \bigcup_{gr}(D^g_{\text{ele}})\), e.g., for TT is \(D_{TT} = \bigcup_{gr}(D^g_{TT})\).

The real-world instants are represented by a granule according to the chosen granularity, e.g., using a day as granularity, a granule \(g_{\text{day}} = 02/10/2006\) represents the
specific day. The time interval is defined as a time between two instances called \textit{begin} and \textit{end} instants, i.e., \([t_{\text{begin}}, t_{\text{end}}]^{gr}\), e.g., [25/09/2006, 02/10/2006]. Thus, the time interval is a sequence of consecutive granules between the starting \((t_{\text{begin}})\) and ending \((t_{\text{end}})\) granules with granularity \(gr\), e.g., all days between 25/09/2006 and 02/10/2006. We also denote \([t_{\text{begin}}, t_{\text{end}}]^{gr}_{\text{VT}}\) the interval for each temporal types, e.g., for VT it is: \([t_{\text{begin}}, t_{\text{end}}]^{gr}_{\text{VT}}\).

We use a temporal element to represent a set of instants or intervals. More formally, a temporal element over time domains is a finite union of intervals, i.e., \(TE^{gr} = [t_{\text{begin}}, t_{\text{end}}]^{gr} \cup \ldots \cup [t_{\text{begin}}, t_{\text{end}}]^{gr}\). We denote \(TE_{\text{VT}}^{gr}\), \(TE_{\text{TT}}^{gr}\), and \(TE_{LS}^{gr}\) as temporal elements of valid-time, transaction-time, and lifespan domains, respectively. Since our time domains are discrete and finite, we can define a temporal element as an element of the powerset \(\mathcal{P}(D_{\text{state}}^{gr})\).

### C.5.2 Semantic domains

The temporally-extended MultiDimER model includes the following value domains:

\begin{align*}
D_{\text{a}} &= \{\perp\} - \text{the set of surrogates} \\
D_{\text{l}} &= D_{\text{a}} - \text{the set of surrogates assigned to } L \in \text{Lev\_TYPE} \\
D_{\text{LS}} &= \bigcup_{gr}(D_{\text{LS}}^{gr}) \cup \{\perp\} - \text{the lifespan domain} \\
D_{\text{VT}} &= \bigcup_{gr}(D_{\text{VT}}^{gr}) \cup \{\perp\} - \text{the valid time domain} \\
D_{\text{TT}} &= \bigcup_{gr}(D_{\text{TT}}^{gr}) \cup \{\perp\} - \text{the transaction time domain} \\
D_{\text{DWLT}} &= \bigcup_{gr}(D_{\text{DWLT}}^{gr}) \cup \{\perp\} - \text{the data warehouse loading time domain} \\
D[\text{DATA}] &= \text{the set of basic domains}
\end{align*}

### C.5.3 Semantic functions

In what follow we describe the signature of the semantic functions and we define the semantic functions.

\begin{align*}
\mathcal{I}C : & \text{Lev\_TYPE} \rightarrow D_{\text{l}}^{gr} \\
T : & T\_SPEC \rightarrow D_{\text{VT}} \cup D_{\text{TT}} \cup D_{\text{LS}} \cup D_{\text{DWLT}} \\
A : & \text{Attributes} \times \text{DATA} \times T\_SPEC \rightarrow \mathcal{D}[\text{DATA}] \cup (T[T_S] \rightarrow \mathcal{D}[\text{DATA}]) \\
L : & \text{Lev\_TYPE} \times T\_SPEC \times \text{Att\_DECL} \rightarrow \mathcal{I}C[L] \times A[A_D] \cup (T[T_S] \rightarrow \mathcal{I}C[L]) \times A[A_D] \\
CP : & \mathcal{P}(\text{Lev\_TYPE}) \times \mathcal{P}(\text{Lev\_TYPE}) \times \text{DATA} \times T\_SPEC \rightarrow \\
& (\mathcal{P}(\mathcal{I}C[L]) \times \mathcal{P}(\mathcal{I}C[L])) \cup (\mathcal{P}(\mathcal{I}C[L]) \times \mathcal{D}[\text{DATA}]) \cup \\
& (\mathcal{P}(\mathcal{I}C[L]) \times T[T_S]) \cup (\mathcal{P}(\mathcal{I}C[L]) \times \mathcal{P}(\mathcal{I}C[L]) \times \mathcal{D}[\text{DATA}] \times T[T_S]) \\
SR : & \text{FR\_TYPE} \times \text{Lev\_Inv\_SPEC} \times \text{Att\_DECL} \rightarrow (\mathcal{I}C[I_L] \times A[A_D]) \cup I[L[I_L]] \\
C : & IC_D \rightarrow \text{PRED} \\
S : & S_D \rightarrow S[S_D]
\end{align*}

The purpose of the semantic function \(\mathcal{I}C\) is to determine the surrogate sets of the level types that are involved in a specific relationship type, i.e., in a fact relationship type or in a child-parent type for the splitting and joining levels of generalized hierarchies. For the former, the definition may include a distributing factor, temporal support, or both. When the distributing factor is present, the function \(\mathcal{I}C\) returns
Additionally the value domain of this factor, i.e., the set of real numbers. When temporal support is included, the changes to members' involvement should be kept. The members are identified through their surrogates, therefore, the timestamps related to the specific temporal support must be associated with members' surrogates.

\[ \mathcal{L}[L] = \begin{cases} D^L_S & \text{if } L \in \text{Lev.TYPE} \\ \bot & \text{otherwise} \end{cases} \]

\[ \mathcal{L}[L \text{ with } \text{disfac}] = \begin{cases} (D^L_S, \mathcal{D}[\text{DATA}](\text{disfac})) & \text{if } L \in \text{Lev.TYPE} \land \text{disfac} \in \mathbb{R} \\ \bot & \text{otherwise} \end{cases} \]

\[ \mathcal{L}[L \text{ during } T_S] = \begin{cases} T[T_S] \rightarrow D^L_S & \text{if } L \in \text{Lev.TYPE} \\ \bot & \text{otherwise} \end{cases} \]

The semantic function \( \mathcal{T} \) is defined in order to determine the time domains of the specified temporal support for a given level type, child-parent type, or attribute.

\[ \mathcal{T}[T_S] = T[T_S] \times T[T_S] \]

\[ \mathcal{T}(t_{ele}, \text{instant, gr}) = D^G_{t_{ele}} \]

\[ \mathcal{T}(t_{ele}, \text{temporal element, gr}) = \mathcal{P}(D^G_{t_{ele}}) \]

The following interpretation functions require additional definitions: the function \( \mathcal{A} \) associates a set of objects \( X = \{X_1, X_2, \ldots, X_n\} \) with the set of value domains \( D \) \( (\mathcal{dom} : X \rightarrow D) \). The instances (tuples) are represented as a set of functions. The domain of each function \( t \) is a set of objects \( X \). Each function \( t \) associates the specific object \( X_i \) with a value from the value domain \( D_i \). We denote it as \( t[X_i] : X_i \rightarrow \mathcal{dom}(X_i) \).

The semantic function \( \mathcal{L} \) identifies the set of functions \( t \) (tuples) that are specified through the definition of level types. The domain of each function \( t \) is the set of attribute names belonging to the level type \( L \) and the surrogate attribute \( s \). The value domain of the surrogate attribute \( s \) is the set \( D^L_S \) of surrogate values assigned to the level \( L \) while the value domain of the attributes of the level \( L \) is determined by the semantics of the attribute declarations.
If the level has defined temporal support, this means that users want to store in the database either lifespan, transaction time, or both for the members of the level type. Recall that lifespan indicates the time that the corresponding real-world member exists while the transaction time refers to the time during which the member was current in the database. Therefore, the timestamps recording the lifespan and/or transaction time must be associated with the member’s surrogate. This is represented as a function from time domain to a set of surrogate values.

The function \( \mathcal{L} \) applied to a composition of level type definitions returns the disjoint union of the functions applied to each of the component. This is since each level type defines a unique set of tuples, which is stored separately in the database.

\[
\mathcal{L}[L_{D_1}; L_{D_2}] = \mathcal{L}[L_{D_1}] \cup \mathcal{L}[L_{D_2}]
\]

\[
\mathcal{L}[\text{Level type } L \text{ has } A_D] = \{t \mid t \in TF \land \text{dom}(t) = \{s, attOfLev(L_D)\} \land t[s] \in D_t^L \land \forall A_i \in attOfLev(L_D) \{t[A_i] \in A[\text{Attribute } A \text{ of type } d]\} \lor t[A_i] \in A[\text{Attribute } A \text{ of type } d \text{ with } T_S]\}
\]

\[
\mathcal{L}[\text{Level type } L \text{ with } T_S \text{ has } A_D] = \{t \mid t \in TF \land \text{dom}(t) = \{s, attOfLev(L_D)\} \\
\land t[s] \in T[T_S] \rightarrow D_t^L \land \forall A_i \in attOfLev(L_D) \\
\{t[A_i] \in A[\text{Attribute } A \text{ of type } d]\} \lor t[A_i] \in A[\text{Attribute } A \text{ of type } d \text{ with } T_S]\}
\]

The purpose of the semantic function \( CP \) is to determine a set of relationships existing between members of the level types specified in the declaration. Members are identified through their surrogates with the value domain defined by \( IL \).

The function \( CP \) is applicable to all types of child-parent type definitions. If the definition of this relationship includes a distributing factor, the domain of the function \( t \) includes additionally an attribute representing it \( (p) \); its value domain is a set of real numbers. If the declaration of the child-parent type refers to the splitting level (respectively, joining) of generalized hierarchies, the set of returned tuples contains surrogate values of members belonging to different parent (respectively, child) level types. If the temporal support is present, it determines that the changes in the relationship between members of child and parent level types must be kept. This relationship is identified through members’ surrogates, therefore, the timestamps related to the specific temporal support must be associated with members’ surrogates.

Since each child-parent type defines a unique set of tuples, if the function \( CP \) applies to a composition of child-relationship type definitions, it returns the disjoint union of the functions applied to each of the component.

\[
CP[CP_{D_1}; CP_{D_2}] = CP[CP_{D_1}] \cup CP[CP_{D_2}]
\]

\[
CP[\text{ChildParent type } CP \text{ relates } L_1 \text{ and } L_2] = \\
\{t \mid t \in TF \land \text{dom}(t) = \{(s_{L_1}, s_{L_2})\} \land t[s_{L_1}] \in IL[L_1] \land t[s_{L_2}] \in IL[L_2]\}
\]

\[
CP[\text{ChildParent type } CP \text{ relates } L_1 \land L_2 \land \text{disfac}] = \\
\{t \mid t \in TF \land \text{dom}(t) = \{(s_{L_1}, s_{L_2}, p)\} \land t[s_{L_1}] \in IL[L_1] \land t[s_{L_2}] \in IL[L_2] \land t[p] \in D[DATA]\}\]

\[
CP[\text{ChildParent type } CP \text{ relates } L_1 \land L_2 \land \text{Ts}] = \\
\{t \mid t \in TF \land \text{dom}(t) = \{(s_{L_1}, s_{L_2})\} \land t[s_{L_1}] \in T[T_S] \rightarrow IL[L_1] \land t[s_{L_2}] \in T[T_S] \rightarrow IL[L_2]\}
\]

\[
CP[\text{ChildParent type } CP \text{ relates } L_1 \land L_2 \land \text{Ts} \land \text{disfac}] = \\
\{t \mid t \in TF \land \text{dom}(t) = \{(s_{L_1}, s_{L_2}, p)\} \land t[s_{L_1}] \in T[T_S] \rightarrow IL[L_1] \land t[p] \in D[DATA]\}\]
APPENDIX C. FORMALIZATION OF THE TEMPORAL MULTIDIMER MODEL

\[ t[s_L] \in T[T_S] \rightarrow \mathcal{IL}[L] \land t[p] \in T[T_S] \rightarrow D[DATA](disfac) \]

\[ CP[ChildParent type CP relates L as splitting level and ILs] = \{ t \mid t \in TF \land \text{dom}(t) = \bigcup_{L_i \in ILs} \{ (s_L, t_L) \} \land \forall L_i \in ILs \}
\]

\[ t[s_L] \in T[tempSpec(ChildParent type CP relates L as splitting level and ILs)] \rightarrow \mathcal{IL}[L] \land
\]

\[ t[s_L] \in T[tempSpec(ChildParent type CP relates L as splitting level and ILs)] \rightarrow \mathcal{IL}[L] \}

\[ CP[ChildParent type CP relates ILs and L as joining level] = \{ t \mid t \in TF \land \text{dom}(t) = \bigcup_{L_i \in ILs} \{ (s_L, s_L) \} \land \forall L_i \in ILs \}
\]

\[ t[s_L] \in T[tempSpec(ChildParent type CP relates ILs and L as joining level)] \rightarrow \mathcal{IL}[L] \land
\]

\[ t[s_L] \in T[tempSpec(ChildParent type CP relates ILs and L as joining level)] \rightarrow \mathcal{IL}[L] \}

The semantic function \( FR \) identifies the tuples specified through the definition of the fact relationship type \( FR \). It determines a set of relationships existing among members from different level types specified by \( ILs \). Members are identified through their surrogates with the value domain defined by \( \mathcal{IL} \). If the declaration of fact relationship type \( FR \) contains attributes, the domain of the function \( t \) includes additionally the set of attribute names. The value domains of these attributes are determined by the semantics of the attribute declarations.

As for the level and the child-parent types, a fact relationship type defines a unique set of tuples that are stored separately in the database.

\[ FR[FRD_1; FRD_2] = FR[FRD_1] \cup FR[FRD_2] \]

\[ FR[Fact relationship type FR involves ILs] = \{ t \mid t \in TF \land \text{dom}(t) = \bigcup_{L_i \in ILs} \{ (s_L, t_L) \} \land \forall L_i \in ILs \}
\]

\[ \forall L_i \in attOfFR(FRD_1) \cup attOfFR(FRD_2) \}
\]

\[ \forall L_i \in ILs \land \{ t[s_L] \in \mathcal{IL}[L] \}
\]

The semantic function \( C \) ensures that the designed database is valid. It guarantees that the constructs mentioned in the constraint declaration are in the schema definition. It also defines the predicates that the objects involved in the constraint have to satisfy. The result of the application of a \( C \) function to a set of constraints is a set of predicates that the database must satisfy.

In the textual representation, all constraints are separate constructs, therefore, for each of them we first should check whether different elements mentioned in the constraint, e.g., level types, child-parent types, exist at all.

\[ C[IC_D_1; IC_D_2] = C[IC_D_1] \land C[IC_D_2] \]

- Primary key constraint.

This constraint ensures that the values of the key attributes are unique for the level set.

\[ C[K is primary key of L] = \text{inSch}(L, S_D) \land \]

\[ K \in \text{attOfLev}(Level \ type \ L \ has \ A_D) \land \]

\[ \forall t_i, t_j \in L \ [Level \ type \ L \ has \ A_D] \ (t_i[K] = t_j[K] \rightarrow t_i[s] = t_j[s]) \]
• A child member can be related to minimum \( \text{min} \) and maximum \( \text{max} \) parent members over its lifespan.

First, we take into account different kinds of child-parent type declarations and refer to the level that plays the role of child. Then, we apply the function \( \text{cnt} \) to determine the number of tuples in the set of members of the child-parent type that member \( m \) of the child level type \( L \) participates in. Finally, this number is compared with the given minimum and maximum values.

\[
\text{in} \{\text{Sch}(\text{min})\} = \text{in} \{\text{Sch}(\text{max})\}
\]

\[
\text{CP} = \text{ChildParent type}\text{ CP relates } L \text{ and } L'
\]

\[
\text{IL}_L \text{ and } L' \text{ as joining level } \wedge \text{IL}_L \text{ and } L' \text{ as joining level } \wedge L \in \text{IL}_L \text{ and } L' \in \text{IL}_L \text{ and } \text{tempLink}(\text{IL}_L) \wedge \forall m \in D_L \text{ (} \text{min} \leq \text{cnt}(m, L, \text{CP}[\text{IL}_L]) \text{ < max)}
\]

• A parent member can be related to minimum \( \text{min} \) and maximum \( \text{max} \) child members during a snapshot time.

First, we take into account different kinds of child-parent type declarations and refer to the level that plays the role of child. We distinguish two groups: one related to the child-parent declarations without temporal support \( (\text{CP}_D) \) and another one related to the declarations with temporal support \( (\text{CP}_D^T) \).

For the first group, we apply the function \( \text{cnt} \) to determine the number of tuples in the set of members of the child-parent type that member \( m \) of the child level type \( L \) participates in. Then, we compare this number with the given minimum and maximum values.

For the second group, we follow the same procedure as for the first group applying it for every granule defined by the specific temporal support.

\[
\text{CP}_D = \text{ChildParent type}\text{ CP relates } L \text{ and } L'
\]

\[
\text{IL}_L \text{ and } L' \text{ as joining level } \wedge \text{IL}_L \text{ and } L' \text{ as joining level } \wedge L \in \text{IL}_L \text{ and } L' \in \text{IL}_L \text{ and } \text{tempLink}(\text{IL}_L) \wedge \forall m \in D_L \text{ (} \text{min} \leq \text{cnt}(m, L, \text{CP}[\text{IL}_L]) \text{ < max)}
\]

• A parent member can be related to minimum \( \text{min} \) and maximum \( \text{max} \) child members over its lifespan.

In order to determine the predicates for this constraint, we apply the same steps as
explained for the similar constraint that refers to a child member. The difference consists in considering now the level that plays the role of parent instead of child.

\[C[\text{Lifespan participation of } L \text{ with } L' \text{ in } CP \text{ is } (\min, \max)] =\]
\[\text{inSch}(L, S_D) \land \text{inSch}(L', S_D) \land \text{inSch}(CP, S_D) \land\]
\[(CP_D = \text{ChildParent type } CP \text{ relates } L' \text{ and } L) \lor\]
\[CP_D = \text{ChildParent type } CP \text{ relates } L' \text{ as splitting level and}\]
\[IL_S \land L \in IL_S \land L \notin \text{tempLink}(IL_S) \lor\]
\[(CP_D = \text{ChildParent type } CP \text{ relates } IL_S \text{ and } L \text{ as joining level } \land L' \in IL_S) \land\]
\[\forall m \in D_{S}^L (\min \leq \text{cnt}(m, L, CP[CP_D]) \leq \max)\]

- A parent member can be related to minimum \(\min\) and maximum \(\max\) child members during time snapshot.

In order to determine the predicates for this constraint, we apply the same steps as explained for the similar constraint that refers to a child member. The difference consists in considering now the level that plays the role of parent instead of child.

\[C[\text{Snapshot participation of } L \text{ with } L' \text{ in } CP \text{ is } (\min, \max)] =\]
\[\text{inSch}(L, S_D) \land \text{inSch}(L', S_D) \land \text{inSch}(CP, S_D) \land\]
\[(CP_D = \text{ChildParent type } CP \text{ relates } L' \text{ and } L) \lor\]
\[CP_D = \text{ChildParent type } CP \text{ relates } L' \text{ as splitting level and}\]
\[IL_S \land L \in IL_S \land L \notin \text{tempLink}(IL_S) \lor\]
\[(CP_D = \text{ChildParent type } CP \text{ relates } IL_S \text{ and } L \text{ as joining level } \land L' \in IL_S) \land\]
\[\forall m \in D_{S}^L (\min \leq \text{cnt}(m, L, CP[CP_D]) \leq \max)\]

- A member of a splitting level is related to members of at most one parent level during time snapshot.

First, we consider the child-parent type declaration related to the splitting level. Then, for every granule defined by the specific temporal support and for every pair of surrogates belonging to the set of members of this child-parent type we check the following: if the member of the child level type is related to a member of some parent level type, there is not another parent level type with a member that is related to the same child member.

\[C[\text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } IL_S] =\]
\[\text{inSch}(L, S_D) \land \forall m \in D_{S}^L \forall L_i \in IL_S \forall m_i \in D_{S}^{L_i}\]
\[(\text{inSch}(L_i, S_D) \land \forall g \in T[\text{tempSpec}(\text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } IL_S)]) \land\]
\[\forall (m, m_i) \in CP[\text{ChildParent type } CP \text{ relates } L \text{ as splitting level and } IL_S] \rightarrow\]
\[\neg (\exists L_j \in IL_S \exists m_j \in D_{S}^{L_j} (L_i \neq L_j) \land\]
\[(m, m_j) \in CP[\text{ChildParent type CP relates } L \text{ as splitting level and } IL_S])\]

The last of the semantic function \(S\) specifies that the MultiDimER schema is composed of definitions of level types, child-parent types, fact relationship types, and integrity constraints. It defines each component of the underlying database and predicates that ensure validity and consistency of the database.

\[S[S_d] = S[L_D; CP_D; FR_D; IC_D] \]
\[S[L_D; CP_D; FR_D; IC_D] = L[L_D] \cup CP[CP_D] \cup FR[FR_D] \cup IC[IC_D] \]
Appendix D

Grapical notations

In the following we include the summary of different graphical notations used in this thesis.

D.1 Notations for the ER model

Since there is not a standard notation for representing the ER construct, we adapt the following symbols\(^1\) for the constructs that we refer in this thesis:

\[^1\]This notation is inspired from [160]
D.1. NOTATIONS FOR THE ER MODEL

cardinalities
(0,n)
Participate
(1,n)
name
relationship type
attributes
Start date
End date
Salary
Week hours
Relationship type
with attributes

Customer
Customer id
Customer name
Customer address
Branch name
Area name

supertype

Person
Profession name
Class name
Specific attributes

type

Company
Type name
Sector name
Specific attributes

Generalization/specialization
relationship type
D.2 Notations for relational and object-relational databases

- **Relational table** (with attributes and keys shown)
  - **Product**
    - Product key
    - Product number
    - Product name
    - Description
    - Size
  - **Category**
    - Category key
    - Category name
    - Description
    - Department key

- **Referential integrity**
  - **Product**
    - Product key
    - Product number
    - Product name
    - Description
    - Size
  - **Category**
    - Category key
    - Category name
    - Description
    - Department key

- **Relational table with instances**
  - **Product**
    - Key
    - Number
    - Name
    - Category key
  - Instances

- **Object-relational table with instances**
  - **Product**
    - Skil
    - Number
    - Value
    - VT
  - Instances

<table>
<thead>
<tr>
<th>Skil</th>
<th>Number</th>
<th>Value</th>
<th>VT</th>
<th>Category key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Q8676</td>
<td>10</td>
<td>05/2002</td>
<td>NOW</td>
</tr>
<tr>
<td>2</td>
<td>QD555</td>
<td>18</td>
<td>05/2002</td>
<td>NOW</td>
</tr>
</tbody>
</table>
D.3 Notations for conventional data warehouses

Members (instances) of different hierarchy levels

<table>
<thead>
<tr>
<th>Level with attributes and keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>key attribute for aggregation</td>
</tr>
<tr>
<td>descriptive attributes</td>
</tr>
<tr>
<td>Product number</td>
</tr>
<tr>
<td>Product name</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Size</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Cardinalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>department A</td>
</tr>
<tr>
<td>category 1</td>
</tr>
<tr>
<td>product A</td>
</tr>
<tr>
<td>category 2</td>
</tr>
<tr>
<td>product B</td>
</tr>
<tr>
<td>product C</td>
</tr>
<tr>
<td>product D</td>
</tr>
</tbody>
</table>

Members (instances) of different hierarchy levels
APPENDIX D. GRAPICAL NOTATIONS

- Non-strict hierarchy
- Fact relationship
  (short description)
- Fact relationship with measures
D.4 Notations for spatial data warehouses

<table>
<thead>
<tr>
<th>Spatial data types</th>
<th>Topological relationship types</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Point</strong></td>
<td><strong>Adjacent</strong></td>
</tr>
<tr>
<td><strong>Line</strong></td>
<td><strong>Intersect</strong></td>
</tr>
<tr>
<td><strong>Surface</strong></td>
<td><strong>Disjoint</strong></td>
</tr>
<tr>
<td><strong>Point bag</strong></td>
<td><strong>Crass</strong></td>
</tr>
<tr>
<td><strong>Line bag</strong></td>
<td><strong>Within</strong></td>
</tr>
<tr>
<td><strong>Surface bag</strong></td>
<td><strong>Equal</strong></td>
</tr>
</tbody>
</table>

**Spatial data types**
- Name: State
- Spatial level (short description)
  - Key attribute for aggregation: State name
  - Thematic attributes: State population, State area, State major activity
  - Spatial attributes: Capital

**Spatial level with attributes shown**

**Spatial hierarchy**

**Spatial fact relationship** (short description)
- Name: Highway
- Spatial calculated measure: Length (S)
- Spatial measure: Condition
D.5 Notations for temporal data warehouses

<table>
<thead>
<tr>
<th>VT</th>
<th>Valid Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>Transaction Time</td>
</tr>
<tr>
<td>BT</td>
<td>Bitemporal Time</td>
</tr>
<tr>
<td>LS</td>
<td>Lifespan</td>
</tr>
<tr>
<td>LT</td>
<td>Lifespan and Temporal Time</td>
</tr>
<tr>
<td>DWLT</td>
<td>Data Warehouse Loading Time</td>
</tr>
</tbody>
</table>

Temporal types

- **temporal type of** the level
- **temporal type of** the level
- **temporal type of** the attributes

Temporal level (short description)

- **name**
- **non-temporal attributes**
- **temporal attributes**

Temporal level with attributes and key

Temporal hierarchy

- **name**
- **Fact relationship**
- **temporal type of the measures**

Fact relationship with temporal measures
Bibliography


251


BIBLIOGRAPHY


