Part I

Background Technologies

Chapter 2

Multidimensional Databases

mOLAP research involves two different technologies: multidimensional databases and data management in wireless, mobile networks. A thorough presentation of these two areas is beyond the scope of this document. Instead of that, this chapter focuses on their properties and characteristics, which are directly thesis relevant.

The chapter is structured as follows: Section 2.1 describes general concepts of the multidimensional space. In Section 2.2, we present the fundamentals of multidimensional models. Section 2.3 presents the data cube operator, while in Section 2.4 and Section 2.5, we elaborate on the two fundamental types of aggregation lattices and the derivation possibilities between their nodes, respectively. Section 2.6 explains the process of query mapping. Finally, in Section 2.7, we underline the distinctive characteristics of multidimensional, compared to relational, querying.

2.1 Multidimensional Space

Data that can be conceptually viewed in a multidimensional space, where each dimension represents a data attribute, is referred to as multidimensional databases (MDDBs). Naturally, there are many more definitions. In *MDDBs*, a data object can be represented as a point in a multidimensional space. For many applications, viewing data in this form is natural and intuitive. Although this might sound weird, even tabular data, such as relations, can be thought of as multidimensional (tables). For example, consider Table 2.1, which represents a three dimensional sales relation with four attributes: *sales (prodID, storeID, timeID, sales)*. Figure 2.1 delivers a multidimensional view of the relation.

There are several reasons for which a multidimensional view of data might be preferable. The most important ones are:

1. **Summarization**: It is probably the most significant reason for viewing data multidimensionally. In databases used for decision support, summary data is used in order to extract meaningful information. Since the amount of data

Table 2.1: Fact table example

prodID	storeID	timeID	sales
81	10	12	15789
78	13	8	13555
34	31	13	578
11	32	13	213
35	80	22	78956
23	88	4	87768

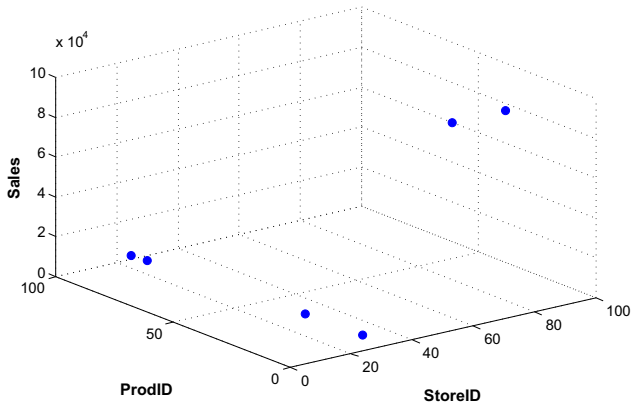

Figure 2.1: Multidimensional view of Table 2.1

substantially grows over time, decision makers have to analyze aggregated data.

2. **Clusters**: In multidimensional space, it is natural to view or even seek clusters. By simply plotting the multidimensional space, such clusters can be identified more easily. Even in the trivial example of Fig. 2.1, it is easy to distinguish 3 minor clusters.

3. **Hierarchies**: A system may contain too many details for a single abstraction level to be intellectually manageable. Thus, dimensions can have category hierarchies associated with them. The capability of accessing information at different levels of abstraction is of major significance. In this sense, summarization can take place over any hierarchical level and not over the dimension.

The term *MDDB* refers to two kinds of databases: Statistical databases (SDBs) and OLAP databases. They have completely different origins. SDBs have a socioeconomic application field, while OLAP a business application field. Apart from the different emphasis given to the data usage and the research done (privacy in SDBs, while efficiency and data analysis in OLAP), another distinction is that

SDBs are usually derived from other base data, while OLAP databases typically directly represent the base data.

Elaborating on the similarities and differences between statistical and OLAP databases is beyond the scope of this document (a more thorough discussion can be found in [146]). In the context of the thesis, it is imperative to underline the fact that both of them handle multidimensional data sets and are concerned with statistical summarizations over the dimensions of the data sets. Consequently, although this work is primarily motivated and intended for OLAP databases, the concepts and solutions introduced are applicable for SDBs as well.

2.2 Conceptual Multidimensional Models

In DWs, in contrast to other application domains, the way in which end users view the information profoundly influences data representation, not only at the physical and logical level, but at the conceptual level as well. Already during the first stages of data warehousing evolution, it was realized that traditional conceptual database models, such as the entity-relationship model, do not provide a suitable description of the fundamental aspects of such applications. Thus, a plethora of multidimensional data models has been proposed. Unfortunately, there is still no consensus on formalism or terminology.

Our work does not assume a specific multidimensional model. Throughout this document we assume a general multidimensional model that includes the two widely recognized entities of any multidimensional schema: the *fact table* and the *dimension*. The fact table is the subject of decision-oriented analysis. It usually consists of the measurements, metrics or facts of a business process, and is represented by means of a data cube. Dimensions correspond to a perspective under which facts can be meaningfully grouped and analyzed. Thus, in retail business for instance, a fact is "sales" and possible dimensions are the location of the sale, the type of product sold, and the time of the sale.

Practitioners usually model these notions using structures that refer to the application's practical implementation. Indeed, a widespread notation used in this context is the *star schema*, in which facts and dimensions are simply relational tables connected in a specific way. Such an example is given in Fig. 2.2. Clearly, this low-level point of view barely captures the essential aspects of the application. Conversely, in conceptual models these concepts would be represented in abstract terms. This is fundamental for concentration on the basic, multidimensional aspects that can be employed in data analysis, as opposed to getting distracted by the implementation details.

Conceptual models can be divided into three main categories:

- **Cube models**: Simple cube models [52, 58, 156] treat data in the form of D-dimensional cubes. They all have a more or less explicit notion of fact, measure and dimension. However, the hierarchy between the various levels of aggregation in dimensions is not explicitly captured by the schema.

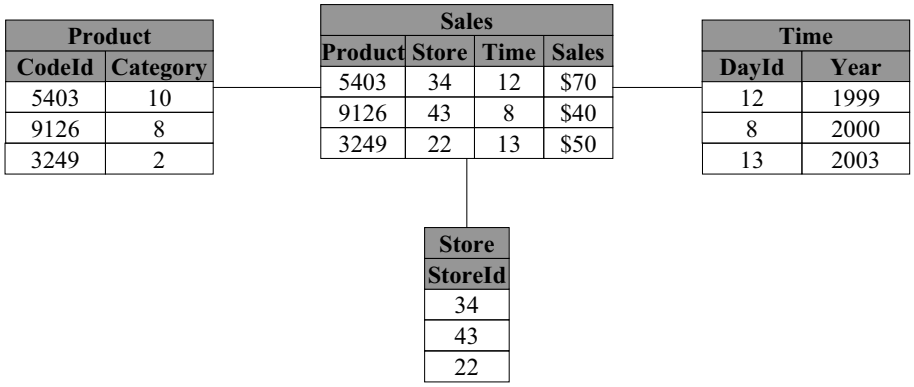

Figure 2.2: Star schema

- **Multidimensional models**: Multidimensional models [11, 159, 76] explicitly capture the hierarchies in the dimensions, providing a better understanding of the application and support for easy data cube manipulation. This information may also be useful for query formulation and optimization.
- **Statistical models**: Statistical data models are usually based on the notions of summary table (ST), summary attribute and category attribute. Actually, there is a close correspondence between these notions and the concepts used in multidimensional data models. Specifically, a ST essentially corresponds to a data cube, a summary attribute to a measure, and a category attribute to a dimension. As in multidimensional models, a category attribute is always associated with a hierarchy of concepts. A number of operators are usually introduced in order to manipulate, concatenate and aggregate STs [130, 146].

2.3 Data Cube

Operators constitute an essential part of data models. A fundamental operator of multidimensional data models is the *data cube* operator, [52, 53, 58, 115]. The data cube is the union of all possible *group-by* operators applied on a fact table. In statistics, this structure is known as *multi-way table*. A data cube stemming from a schema with D dimensional attributes has 2^D possible sub-cubes. Figure 2.3 depicts the data cube produced by the data of Table 2.1 and 2 sub-cubes derived from this cube.

According to the theory proposed in [101]:

Definition Given for a type t and its domain $dom(t)$:

1. A grouping L_i is defined on a set of selection expressions $\sigma_{a_{i,1}}, ..., \sigma_{a_{i,n_i}}$.
2. The grouping L_i is finer than L_j if either $a_{i,k} \rightarrow a_{j,l}$ or $(a_{i,k} \wedge a_{j,l}) \leftrightarrow 0$ for all $k(1 \leq k \leq n_i)$ and $l(1 \leq l \leq n_j)$. The trivial grouping is denoted by ALL.

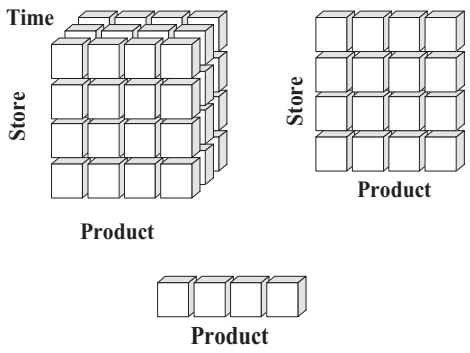

Figure 2.3: Data cube

3. The grouping L_i is a refinement of the grouping $L_j \preceq (L_i \preceq L_j)$ if each group $g_{i,k}$ is subset of exactly one group $g_{j,l}$. In this case, an anchoring function anc^{L_i,L_j} and a relation $desc^{L_i,L_j}$ that is inverse to anc^{L_i,L_j} are defined for each pair $L_i \preceq L_j$.

4. A hierarchically ordered dimension D consists of a type and a set of groupings $(\{L_1^D, ..., L_n^D, ALL\}, \preceq)$ that form a lattice.

5. Hierarchically ordered dimensions are well defined if all groupings form partitions (are pairwise disjoint and form a cover).

According to [100], only well-defined hierarchical dimensions are considered. The time dimension is a typical example of a dimension. Used are types $Seconds$, $Minutes$, $Hours$, $Days$, $Weeks$, $Months$, $Years$ and the linear partial orders $Seconds \preceq Minutes \preceq Hours \preceq Days \preceq Months \preceq Years$, $Days \preceq Weeks$, $Weeks \not\preceq Months$, $Weeks \not\preceq Years$, where the function $anc^{Minutes,Hours}$ maps minutes (e.g., 10:02 am) to the hour they are embedded (e.g., 11 am).

Definition A cube schema $C = (D_1, ..., D_m, M_1, ..., M_k, \Sigma_C)$ is given by:

1. A set of well defined dimensions $\{D_i | 1 \leq i \leq m\}$ that form a key of C.

2. A set of fact attributes $M_1, ..., M_k$, an associated set of aggregation functions F, and a set of associated transformations $t_1, ..., t_k \in T$.

3. A set of integrity constraints Σ_c.

Definition A cube algebra is given by:

1. A cube schema C.

2. An algebra consisting of at least navigation, selection, projection and split functions.

Table 2.2: Declared hierarchies of a 3-dimensional data cube

Hierarchies					
Product		**Store**		**Time**	
ALL	P_0			ALL	T_0
↑	↑	ALL	S_0	↑	↑
Category	P_1	↑	↑	Year	T_1
↑	↑	StoreId	S_1	↑	↑
Code	P_2			Day	T_2

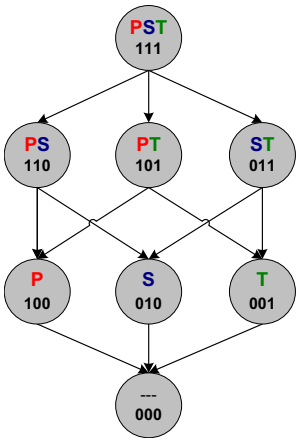

Figure 2.4: DCL of a 3-dimensional data cube

2.4 Aggregation Lattices

This paragraph describes the notion of the *aggregation lattice*. Aggregation lattices provide a visual representation of a data cube, its sub-cubes and the relationships between them. There are two types of aggregation lattices, depending on the inclusion of hierarchical levels of dimensions or not. Table 2.2 contains the declared hierarchical levels of the multidimensional schema of Table 2.1.

Data Cube Lattice

A data cube stemming from a schema with D dimensional attributes has 2^D possible sub-cubes. Given multidimensional data, the Data Cube Lattice (DCL) is the lattice of the set of all possible grouping queries that can be defined on the foreign keys of the fact table [22]. It is a directed, acyclic graph, which depicts the relationships between all 2^D sub-cubes. Assume the 3-dimensional schema of Table 2.1. Figure 2.4 shows the corresponding *DCL*. Each of every possible sub-cube is represented in the lattice by one node.

DCL nodes can be labeled by a sequence of bits (bitmap), as depicted in Fig. 2.4. The number of necessary bits is equal to the dimensionality of the cube. Each bit represents one dimension. If the dimension exists in the node, then the bit is set to 1, otherwise it is set to 0.

Note that the hierarchical levels of each dimension are completely ignored in *DCLs*.

Hierarchical Data Cube Lattice

Hierarchies on aggregation lattices were introduced in [60, 148]. A hierarchical Data Cube Lattice (hDCL) is a directed, acyclic graph, which depicts the relationships between all $\prod_{n=1}^{D}(gr_i + 1)$ sub-cubes, given a D-dimensional cube and the number of grouping attributes gr_i of each dimension.

Note that this definition considers a limited set of grouping attributes. It considers only the dimension's key attribute and the non key attributes that are functionally dependent on it (practically an attribute hierarchy). If the set of grouping attributes includes attributes not functionally dependent on the key attribute as well, then the produced lattice is called *MD-lattice* (Multidimensional lattice), as defined in [22]. *MD-lattices* are not considered in this thesis, since the number of their nodes is so high, that the fundamental objective behind query mapping of reducing the handled data items cannot be fulfilled. Furthermore, the hierarchies are *strict* (not weak [98]), namely each object at a lower level belongs to only one value at a higher level.

It is important to underline that the only key difference between *DCL* and *hDCL* is the degree of detail. Figure 2.5 contains the *hDCL* that corresponds to the schema of Table 2.2. Essentially, the *DCL* is a subset of the respective *hDCL* (gray nodes of Fig. 2.5). There are $(gr_P{+}1){\times}(gr_S{+}1){\times}(gr_T{+}1){=}3{\times}2{\times}3{=}18$ possible views or sub-cubes. Similarly to the bit notation of the *DCL*, we notate *hDCL* nodes with a sequence of digits. Each digit represents the dimension and the hierarchical level. If the dimension does not exist in the node, then the digit is set to 0, otherwise it is set to the number of hierarchical level. For example, the sub-cube P_2S_1 in Fig. 2.5 is marked with 210. The first digit is 2, and indicates the second hierarchical level of dimension *Product*, the second digit is 1 indicating the first hierarchical level of dimension *Store*, and the last digit is 0, indicating that dimension *Time* has been projected.

2.5 Derivability - Subsumption

Consider two queries: q_1 and q_2. q_1 is dependent on q_2 ($q_2 \succeq q_1$) when q_1 can be answered by using the result of q_2. This property is known as *query dependency*. The reuse of queries in *MDDBs* is mainly related to the *data cube* operator [53, 58]. [60] notes that some of the group-by queries in the data cube query can be answered using the results of other. In *MDDBs*, there are two types of query dependencies:

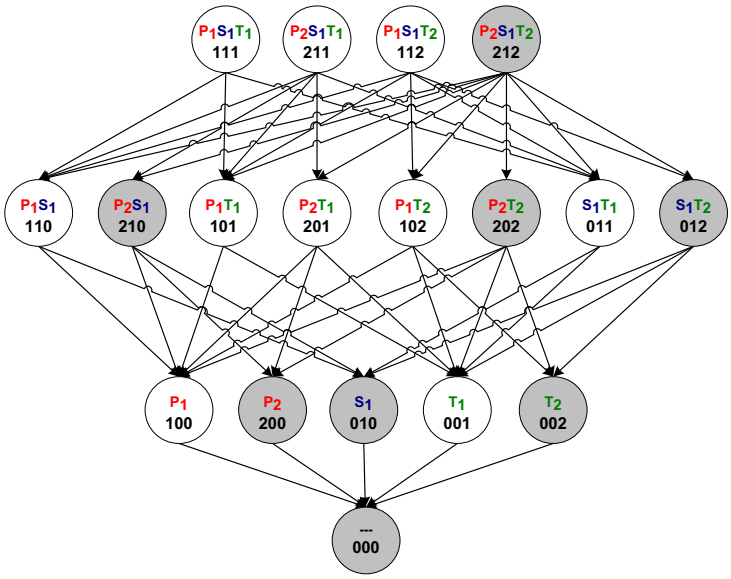

Figure 2.5: hDCL of a 3-dimensional data cube

- **Dimension dependency** is caused by the interaction of the different dimensions with one another (e.g., P_1 dependent on P_2S_1).

- **Attribute dependency** is caused within a dimension by attribute hierarchies (e.g., P_1S_1 dependent on P_2S_1).

Data cubes are created from group-by queries, for which dependencies exist. Consequently, dependencies also exist between the produced sub-cubes. The nodes of an aggregation lattice represent different views of the data cube, whereas its arcs represent *derivability of sub-cubes*.

Derivability is not a new research area. [134] back in 1981 introduced it in the context of statistical databases, by checking derivability of summary data under different classifications. In the following years, derivability became extremely important, both in relational and multidimensional databases, in the context of materialized views [57, 12, 16, 86, 79]. When using materialized views in *MDDBs*, it is critical to find the set of materialization that maximizes the performance in answering a given set of representative queries. The tradeoff consists of choosing a set of materialization able to speed up query response time, while minimizing the overhead to keep materialization updated.

A sub-cube can be derived by another sub-cube, if there is a path in the aggregation lattice that connects the corresponding nodes. This is known as *subsumption*. The term *additivity* is also used for this notion [63]. This derivation is feasible for distributive SQL aggregation functions such as *sum, min, max* or *count*, but is neither allowed for algebraic functions such as *average* or *covariance* nor for holistic functions like *median* [99, 100, 101]. Aggregating over a time di-

Table 2.3: Derivability for aggregate functions

Function	Class	Subsumption
sum	distributive	yes
min	distributive	yes
max	distributive	yes
count	distributive	yes
avg	algebraic	no (yes if count table at hand)
covariance	algebraic	no
median	holistic	no
rank	holistic	no

mension is allowed if the fact (measure) is of type *flow*. Table 2.3 summarizes subsumption feasibility for the most common aggregate functions.

The *ancestor* and *descendant* operators, as defined in [22, 160], also reveal query dependencies. The result of the ancestor operator $\oplus$ on two queries is the smallest query containing all necessary information to answer both queries, whereas the descendant operator $\ominus$ computes the greatest query among the set of attributes characterizing the queries that can be computed by the two queries. In this document though, we do not use the term ancestor as an operator, but as a property to representing sub-cube derivability. In this context, a lattice node n_a is an ancestor of a lattice node n_b, if there is a downward path from n_a to n_b in the lattice.

hDCL arcs represent dimension and/or attribute dependencies, whereas *DCL* arcs represent exclusively dimension dependencies.

2.6 Query Mapping to Aggregation Lattices

Query mapping is the process of mapping a query to its corresponding node in the aggregation lattice. It shall be shown that query mapping is a fundamental component in mOLAP systems. We provide an example of how a query would be mapped in the two discussed lattices. Assume the following query targeting the schema of Table 2.2 and one measure attribute. Without loss of generality, we use Multidimensional Expressions (MDX) as the query language.

```
SELECT
 { [Product].[Category].[Drinks] } ON COLUMNS,
 { [Time].[Year].AllMembers } ON ROWS
FROM [SalesCube]
```

This query is mapped to node PT of the *DCL*, since only dimensions *Product* and *Time* are involved. Concerning the *hDCL*, the query is mapped to node P_1T_1,

since the member *Drinks* belongs to the hierarchical level *Product.Category* (P_1), and all members of the hierarchical level *Time.Year* must be retrieved (T_1).

It is important to underline though that query mapping is not always so straightforward. Selections or clauses might not target the attributes of the fact table only, but the attributes of the dimension tables as well. In this case, the query cannot be answered by the respective lattice sub-cube only, and thus additional dimension table data are necessary.

2.7 Querying Multidimensional Data

In the same way that multidimensional modeling is based on the metaphor of the data cube and on the concepts of facts, measures and dimensions, the techniques to retrieve such data are based on the idea of determining the cube of interest and navigating through it. Multidimensional querying is profoundly different than relational querying. The presence of aggregations is one of the most significant distinctive features of OLAP systems with respect to conventional transactional systems. We restrict our discussion to the characteristics of multidimensional querying that are more thesis relevant:

- **Subsumption**: The answer to a multidimensional query is a sub-cube. According to the subsumption, multiple queries that refer to sub-cubes for which subsumption can be applied, can be answered using the ancestor sub-cube only.

- **Variant sizes**: Each sub-cube occupies a different (physical) size. The size is dependent not only on the cube's dimensionality and the cardinality of its dimensions, but on its physical storage implementation as well. For example, a typical DW might contain a dimension *SEX* with cardinality 2 (male, female) and a dimension *Product* with cardinality 10K. Moreover, the average dataset produced by a multidimensional range query is on average much bigger in size than the one produced by a relational query.

- **Skewness**: Data cubes typically contain hot areas because some sub-cubes are more often requested. Due to the fact that queries refer to dimensions and some dimensions are more popular, data cube areas representing these dimensions are more likely to be queried.

- **End user behavior**: End users typically navigate through the query results, performing typical OLAP operations such as rolling-up, drilling-down, dicing and slicing. In this sense, one sub-cube can be used to answer more than one multidimensional query.

Naturally, the aforementioned properties of multidimensional queries directly influence the design of mOLAP architectures.

2.8 Summary

This chapter gives a brief overview of multidimensional databases. Beyond the presentation of general concepts, our discussion concentrates on the data cube, its operators, aggregation lattices, subsumption between aggregation lattice nodes, as well as on query mapping. There are two types of aggregation lattices, depending on the inclusion of hierarchical levels of dimensions or not.

As to be seen in the following chapters, point to point communication for mOLAP exhibits poor performance. Therefore, mOLAP architectures use query mapping to the corresponding lattice nodes in order to reduce the number of handled data items and exploit subsumptions between them. In mOLAP systems, when mapping queries to lattice nodes, two queries corresponding to different lattice nodes for which a dependency exists, do not have to be served by two separate transmissions, but from a single broadcast instead.