A Survey on On-Line Analytical Processing

Rudolf Strijkers [rjstrijk@science.uva.nl]
16th June 2004

Supervisors: Wilko Hische
Dick van Albada
Inge Bethke

University of Amsterdam
Abstract

This thesis is a comprehensive study of what characterises an On-Line Analytical Processing (OLAP) system. The features of OLAP are explained with the goal to clarify the term OLAP. It concludes with two product descriptions and provides pointers for further exploration of this technology which has been around for a while but is gaining popularity among the masses.
# Table of Contents

Abstract

I

Table of Contents

II

1 Introduction

1.1 Objective of this thesis .................................. 2
1.2 Outline .................................................. 2

2 OLAP Concepts

2.1 Introduction .............................................. 4
2.2 OLAP Requirements ...................................... 4
2.3 Dimensions and Measures ................................ 5
   2.3.1 Hierarchies ......................................... 5
   2.3.2 Ordering ............................................ 7
2.4 Snowflake/Star Schema .................................. 8
   2.4.1 Introduction ....................................... 8
   2.4.2 Example ............................................ 8
   2.4.3 Conclusion ........................................ 11
2.5 Cubes and Hypercubes ................................... 12
2.6 Multidimensional Curse ................................. 13
2.7 Storing Multidimensional Data .......................... 14
   2.7.1 ROLAP ............................................. 15
   2.7.2 MOLAP ............................................. 16
   2.7.3 HOLAP ............................................. 16
   2.7.4 DOLAP ............................................. 16
2.8 Analytic Visualization .................................. 18
   2.8.1 Graphical Representation .......................... 18
   2.8.2 Numerical Representation .......................... 20
   2.8.3 Pivot Tables ....................................... 21

3 OLAP Products .............................................. 24

3.1 Introduction ............................................ 24
3.2 Microsoft SQL Server – Analysis Services ............. 25
   3.2.1 Overview ......................................... 25
Chapter 1

Introduction

On-line Analytical Processing - OLAP - first mentioned in an unpublished white paper by E.F. Codd [4] has been a field of research since the late seventies. The relational model defined by Codd turned out to be not very suitable for data analysis because it was more focused on the low-level detail than on the higher level abstraction that analysts needed.

The basic concept of OLAP is that the world is ordered in a multidimensional hierarchical manner. This means that to be able to do analysis we have to provide tools to navigate through and perform calculations on multidimensional data. This data could be anything and is typically stored in a relational database.

There has been much debate on the functional requirements of OLAP systems. Because there is no generally accepted definition of OLAP and OLAP itself has several different interpretations and different abstraction levels it is impossible to provide a fully comprehensive definition of the term OLAP. Codd was the first to formulate the characterization of OLAP systems that was originally published in Computerworld [4]. A much simpler characterization called FASMI was defined by N. Pendse, publisher of The OLAP Report. This will be explained in the next chapter. See the references for the original document [19].

1.1 Objective of this thesis

This survey on OLAP tries to identify and summarize key features that characterize an OLAP tool. The aim is to gain understanding of the problems that OLAP claims to solve with the ultimate goal to be able to make a good choice between OLAP products based on business needs.

1.2 Outline

Chapter 2 of this thesis introduces the concepts of OLAP beginning with the definition of the requirements of an OLAP system. Then dimensions and meas-
ures – the basic building blocks of a multidimensional model – are explained together with hierarchies that can be defined on dimensions. Ordering is a basic concept returning in all facets of OLAP (from storage to visualization). Nominal, ordinal and cardinal ordering are explained in the light of hierarchies. Section 2.4 explains how to build a warehouse using a star or snowflake schema and how different kind of hierarchies can be constructed in the relational model. Cubes are the basic data structure in every OLAP tool. Section 2.5 discusses two different logical models for representing multidimensional data cubes. When dimensions are added to dense cubes or cubes with a few dimensions in common are joined the resulting cubes can become very sparse. Section 2.6 explains how to detect if a join of cubes or addition of dimensions to a cube is meaningful and illustrates the implications of sparse data. Multi dimensional data can be stored in a number of ways each way having different advantages and disadvantages. Section 2.7 takes a look at the different approaches and sums up the pros and cons.

The analytical visualization section tries to explain that data can be visualized in many ways but that every kind of data has a ‘best representation’. This section also explains that there are rules for determining the best representation for a specific set of data.

The main goal of Chapter 3 is to show a few products to illustrate the concepts presented in chapter 2. The most important features of Mondrian, Applix TM1 and Microsoft Analysis Services are explained. These products have been chosen because Mondrian is the only serious open source OLAP server, TM1 has been the OLAP market leader for long time and was one of the first RAM based OLAP servers and Microsoft which wanted a share of the OLAP market built the first OLAP solution affordable for the masses.

In the conclusion of this thesis some application areas of OLAP are mentioned and it is explained why OLAP a technology to stay.

Chapter 5 contains a view pointers for future work.

Appendix A sums up Codd’s twelve rules from the paper ‘Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate’. The paper turned out to be vendor specific. It was commissioned by Arbor Software (now Hyperion) and the OLAP product of Arbor was the first one to meet all the features mentioned in the paper. These rules are given mainly for historical importance.

Appendix B shows an example of the popular Microsoft Multi Dimensional eXpression language, MDX. This illustrates the use of a query language in an OLAP product.

The last appendix shows a list of the most popular products on the OLAP market. This list is far from complete but can be a starting point for evaluating OLAP products.
Chapter 2

OLAP Concepts

2.1 Introduction

The purpose of this chapter is to provide a solid introduction to OLAP by describing the most important features of an OLAP system. The OLAP world introduces a lot of lingo with many words referring to the same thing. For consistency only a small set of the lingo will be used. Once the terms are clear it should be trivial to understand how those things are referred to. For example a measure is also called a variable and a fact table is often referred to as a base table.

2.2 OLAP Requirements


Fast Speed is crucial to the usability of OLAP systems. Users need the system to respond quickly to maintain their train of thought. This means that ad hoc queries have to execute within no more than two to three seconds or an acceptable limit for very complicated queries.

Analysis An OLAP system has to be flexible. It has to provide rich analytical functions with a minimum of programming, different ways of viewing analytical data and flexible ways of defining views, analysis or data. Separating the structure from the representation gives the user the ability to view analysis data differently.

Shared A practical requirement for OLAP is multi-user support. It is very likely that more users will be wanting to work on the system to perform analysis. Therefore, degrees of access (or access control), integrity and multi-user caching should be covered by the system.
Multidimensional  The most important characteristic of OLAP is multidimensionality. Independent of the underlying storage system the user should have a multidimensional view on the data. There has to be rich support for different kinds of hierarchies and visualization. Navigating through and performing calculations on the hierarchies needs to be flexible and easy.

Information  The amount of data that an OLAP system can deal with is an important characteristic of that system and depends on a lot of factors. For instance, how much of the analytical data is pre-calculated, whether the warehouse data can fit in memory, how disk space is used, e.a.

2.3 Dimensions and Measures

Data used for analysis can be divided in two components. A Measure contains the quantitative values of one kind of data (sales or costs for example). This data is the subject of analysis and is often gathered from real world events (for instance shoe sales in June at some store). Dimensions are categories into which the quantitative data can be grouped. Measures are analysed against dimensions. Dimensions are often ordered in some kind of hierarchical manner.

2.3.1 Hierarchies

A lot of things in the world can be divided in groups or classified in different ways. Time for instance can be divided in years, quarters, months and days or an organization in president, head of department, chiefs and employees. To be useful the groupings should have some kind of ordering defining the relation between the elements in the hierarchy. Taking the time grouping for instance an ordering could be three months in a quarter, four quarters in a year and so on. An OLAP system should be able to model efficiently (or at least support) any type of hierarchy and ordering.

Figure 2.1: A ragged hierarchy
There are basically three types of hierarchies. Ragged hierarchies represent irregularly grouped data like a reporting hierarchy in a company (see figure 2.1). Leveled hierarchies represent grouped data with a specific order defined on it (see figure 2.2). Mixed hierarchies have properties of both the ragged and leveled hierarchy. Levels in a mixed hierarchy have to be fully connected (or: A instance of level x, B instance of level y, B direct under A. For every A at least one B that is a child of A and no children not in B and for every B one parent in A and no parent not in A.)

Not all tools support all types of hierarchies. A ragged hierarchy modelled in a level-based tool or a leveled hierarchy modelled in a ragged-based tool is called a deviant hierarchy. These hierarchies are usually not very efficient.

Navigating from/toward the root of an hierarchy is called drilling down/up.

---

**Figure 2.2: A leveled hierarchy**

It is also possible to have multiple hierarchies defined on a dimension. This allows the dimension to be aggregated in different ways. Figure 2.3 shows a multiple hierarchy.

---

**Figure 2.3: A Multiple hierarchy can roll up in more ways**
2.3.2 Ordering

In the previous subsection it was mentioned that leveled hierarchies have some kind of ordering. Leveled hierarchies can be ordered in a nominal, ordinal or cardinal manner. Every ordering has the same properties of the ordering before but adds one property. Ordering by identity is nominal ordering. The elements of a nominal hierarchy are ordered by kind (therefore only test on equality has meaning). Ordinal ordering is based on rank: first, second and thirth or big, medium and small. Cardinal ordering is ordering by number. Operators as addition, subtraction and multiplication have meaning on this kind of ordering. Examples of nominal, ordinal and cardinal orderings are respectively: cities ordered by region, cities ordered by size, distance ordered by kilometers, meters and centimeters (see figures 2.4, 2.5, 2.6).

Figure 2.4: Example of a nominal ordering

Figure 2.5: Ordinal extension of figure 2.4 with ranks: big, medium and small
2.4 Snowflake/Star Schema

2.4.1 Introduction

A data warehouse is a database - usually multidimensional or relational - that varies in time and is primarily used for analysis based decision making. A data warehouse integrates data from a single or different information source(s) and contains historical and aggregated data \[13\]. Aggregations are precalculated summaries of data that speed up query performance \[6\]. This could be anything between pre-calculated summaries of hierarchies to speed up drilling to aggregating low-level fine grained data to a more usable form usually defining the grain of the smallest element in the model. For example data containing stock price changes per minute are too detailed for analysis on stock price changes over a few years. In this case aggregating to stock price changes per day would significantly speed up query performance without significant loss of information of interest.

Because OLAP data is usually large in quantity it is often necessary to summarize it in some way for humans to be able to derive information of it. The snowflake/star schema is the most common form used for storing multidimensional data in relational databases. See section 2.7 for more information about storing aggregated data.

The basic form, a star schema is a fact table containing the quantitative data with dimension tables directly connected to the fact table. A snowflake schema is a star schema but the dimension tables can have subdimension tables. Cycles are not allowed.

2.4.2 Example

There are many ways to reduce a star/snowflake schema to tables. The following examples will demonstrate a few ways to build up a star and snowflake schema with dimensions having ragged or leveled hierarchies. For in-depth information
about database design and concepts see [1] and [13] for star/snowflake schema specific information.

A ragged hierarchy is modelled in a relational database using two columns which together define a parent-child relationship. Other columns could be added to define attributes of a child. Figure 2.7 shows an employee reporting hierarchy and the corresponding table in figure 2.8.

![Figure 2.7: Reporting hierarchy for employees](image)

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>EMPLOYEE_MANAGER</th>
<th>EMPLOYEE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Jake</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Fred</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Chris</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Lisa</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>John</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>Edward</td>
</tr>
</tbody>
</table>

*Figure 2.8: Corresponding table of figure 2.7*

Leveled hierarchies can be modelled using the level names as columns in the table. There are two ways to model a leveled hierarchy. The first is to put all levels in one table. This introduces a lot of redundancy. Another method can be used in snowflake schemas. The levels are then separated into smaller tables containing only level information with foreign keys to the parent level or child level (or both). This method decreases the redundancy but requires a lot of joins. Figure 2.9 shows the table for figure 2.2 in subsection 2.3.1 for a star schema and figure 2.10 for a snowflake schema.

<table>
<thead>
<tr>
<th>MONTH_ID</th>
<th>MONTH_NAME</th>
<th>QUARTER_ID</th>
<th>QUARTER_NAME</th>
<th>YEAR_ID</th>
<th>YEAR_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>January</td>
<td>1</td>
<td>One</td>
<td>1</td>
<td>2002</td>
</tr>
<tr>
<td>2</td>
<td>February</td>
<td>1</td>
<td>One</td>
<td>1</td>
<td>2002</td>
</tr>
<tr>
<td>3</td>
<td>March</td>
<td>1</td>
<td>One</td>
<td>1</td>
<td>2002</td>
</tr>
<tr>
<td>4</td>
<td>April</td>
<td>2</td>
<td>Two</td>
<td>1</td>
<td>2002</td>
</tr>
<tr>
<td>5</td>
<td>May</td>
<td>2</td>
<td>Two</td>
<td>1</td>
<td>2002</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

*Figure 2.9: Time dimension table in a star schema*
Multiple hierarchies can be built by defining separate tables for every different hierarchy on a dimension.

Figure 2.11 defines a star schema for a manager who wants to track the amount of hours his employees work on several duties and how much rest they take.

Questions like: ‘Can you show me the hours employee X worked in the months January to June with the hours worked on duties?’; ‘What is the ratio between worked hours and breaks for all the employees?’ and ‘Do my managers work longer than their employees?’ can be answered by using SQL but requires a fair amount of joins in the queries and requires views to be created for every possible useful representation of the data in the schema.

Creating the answer to the questions would need the following steps:
1. Join dimension tables with the fact table(s)
2. Create any needed aggregations
3. Create a view that represents the best view on the data
4. Fill the view with data

If one would want to create aggregate views of the levels in the time dimension this would take three fact table views (days, months and years). A star schema with four dimensions and 3, 4, 2 and 4 levels respectively would need 96 aggregate views to provide drilling up/down functionality for the dimensions. Although SQL can model multidimensional data it is a very cumbersome task to provide different views on the data or to perform calculations that are not column-oriented (For example: to compute the ratio of sales and costs that are not stored by column it is necessary to put them in sales and costs columns first before a view can be created that calculates the ratio in the select query.)

2.4.3 Conclusion

The main drawback of providing OLAP functionality with relational databases with the use of star/snowflake schema is that even a simple OLAP query can result in very complex SQL. Also SQL has a lot of other limitations that does not make it very suitable for OLAP use. To enable relational databases to do OLAP-like queries SQL-3 provides analytical extensions. Many database vendors do also provide SQL extensions independent from SQL-3. For more articles on relational extensions for OLAP see [16, 13, 10].


2.5 Cubes and Hypercubes

OLAP data is modelled in cubes or hypercubes. A cube or hypercube is a logical multidimensional structure that contains one or more measures with a set of dimensions categorising the data in a meaningful manner (see 2.12). Cells of the cube are (unique) intersections with a member (item in a dimension) of every dimension representing a value or when more measures are used a set of values. An logical OLAP model can be implemented several ways namely the single domain and multi domain schema.

When putting all measures and dimensions in one single hypercube this is called a single domain schema. Separating related measures and dimensions from not related dimensions and measures in different cubes is called a multi domain schema. These cubes can be joined to form a virtual hypercube when they have dimensions in common.

The first picture of figure 2.13 shows two cubes with two shared dimensions, products and scenario. The four dimensional cube contains two additional dimensions and the other contains three additional dimensions other than the shared dimensions.

A multi cube can then be visualized by a cube (in this example two dimensions) composed of the shared dimensions containing the subsets of the other cubes. This is shown by the lower picture in figure 2.13. The multi cubes can be visualized by displaying the subsets for a specific intersection of the shared dimensions.

![Figure 2.12: A single domain schema](image-url)
2.6 Multidimensional Curse

In the previous section two schemas were defined into which OLAP data can be modelled. This subsection demonstrates what the implications are when a model contains sparse data, that is cells containing meaningless or inapplicable data. An example of sparse cube would be a sales measure that contains the sales of sports products categorised by store and time from year to week while some of the products are only sold in summer and others only in winter.

In a single domain schema a hypercube can become very sparse. To test if a combination of measures and dimensions will become sparse one can look at the domain of the data. If two data sets do not belong to a single hypercube the merge of those two sets will have a lower density than the density of the sets.
Imagine a company selling twenty products at forty stores collecting three measures divided by ten time periods. A hypercube modelling this data will consist of: \(20 \cdot 40 \cdot 3 \cdot 10 = 24000\) cells. Now take another data set with the same store and time dimension but with thirty countries and eight customer features giving: \(40 \cdot 10 \cdot 30 \cdot 8 = 96000\) cells. If these cells are all filled a merge of these cubes will result in: \(40 \cdot 10 \cdot 20 \cdot 3 \cdot 30 \cdot 8 = 576 \cdot 10^4\) cells. Because there are only \(24 \cdot 10^3 + 96 \cdot 10^3 = 12 \cdot 10^4\) data points the combined cube will be approximately 98% empty.

One of the features defined by Codd is to be able to deal with missing or inapplicable data by distinguishing those from zero values and it is vital for the correctness of calculations performing on multidimensional data. When summing sales data by store and time for instance it is important to know whether a missing store sales summation is due to late submission of data so that a projection can be made based on historical data or that the store was closed and no data was available.

![Figure 2.14: Example of database growth](image)

### 2.7 Storing Multidimensional Data

Data to be used for OLAP is usually stored in a relational database or other data source not optimized for multidimensional analysis. Often OLAP tools provide their own optimized data structure(s) for storage. These optimized structures are also known under the term ‘data warehouse’. This section explains different architectures for building data warehouses. It does also explain factors that have to be dealt with before reliable analysis can be done on the data.

The role of a warehouse is to transform and store data usually originating from different sources in a consistent manner to provide fast multidimensional access to the data without compromising the performance of the operational system(s).
delivering the data. It has to take into account factors that can influence the correctness of the data and deal with it accordingly.

Most important factors [17]:

**Consolidate source data** Data coming from different data sources can have very different formats that potentially can not be easily merged into a single representation. Therefore clear schemes for converting the data to a general form have to be made. Take for instance a company with stores located in The Netherlands and Japan. The sales data is stored in Euro’s and Yen respectively. For analysis on company sales the currency should be the same. Also other things like the naming schemes for product id’s at the stores should be converted too.

**Aggregate** Often the source data is too detailed for the high-level decision-making activities and it is therefore common practice to summarise the data to a larger grain size to achieve better performance. The degree to which data is aggregated or pre-calculated highly determines query performance and the system response time. It is impossible to pre-aggregate and pre-calculate all the data considering load time and disk space so it is desired to keep this down to a minimum (see figure 2.14).

**Update warehouse data** Source data can have different intervals on which it is updated. Some data is updated once per week, other at certain events. To keep warehouse data consistent it has to be updated regularly with a well chosen update scheme. Because analysis is often time based it is often a good idea to store historical data in the warehouse too. Analysis on warehouse data may also lead to changes in the source data. Some warehouses support the ability to lead back the changes to the source data.

There are a number of options to store OLAP data namely ROLAP, MOLAP and HOLAP. These abbreviations stand for Relational, Multi dimensional and Hybrid OLAP respectively. DOLAP or Desktop OLAP labels the more low-priced, simple OLAP tools. The next subsections will explore these architectures.

### 2.7.1 ROLAP

ROLAP tools store multidimensional data in relational databases often in denormalized database schemas, mostly a star or snowflake schema or a variation on the theme. A few advantages of this approach are that these tools can often store and process large amounts of data, RDBMS (Relational Database Management System) knowledge is usually at hand and that features like recovery, rollback and replication are already present in the RDBMS. Even the most simple multidimensional query requires complex SQL statements which require a lot of joins. This makes querying the database very slow.
To optimize querying of the relational database the output of queries can be cached in the database or in a multidimensional structure. Although this architecture generally does not have to deal with sparsity of data, caching the intermediate or final result of queries does introduce this issue.

### 2.7.2 MOLAP

A multidimensional database (MDDB) stores data in array like structures to provide optimal access time to every element in the structure. Multi dimensional databases pre-calculate and pre-aggregate data to get fast query response time. This results in longer load time and usually requires considerable amounts of disk space. Because of the pre-calculation of the cells the multidimensional curse (see section 2.14) comes in play. To avoid database explosion it is important that the MDDB has good sparsity handling.

Figure 2.15 was taken from Ovum Evaluates [12] and sums up the pros and cons of the ROLAP and MOLAP architectures.

### 2.7.3 HOLAP

Hybrid OLAP uses both approaches in some kind of way to provide optimized storage for OLAP queries. Generally MOLAP solutions are better when the data is dense and not very large. ROLAP is better at handling large amounts of data. One way hybrid solutions use both approaches is to store aggregated and pre-calculated data in a multidimensional database either in RAM or on disk and keep the detailed data in the relational database.

### 2.7.4 DOLAP

Desktop OLAP tools perform analysis on clients which can range from desktop computers to web browsers using data originating from relational or multidimensional databases that can either reside on the local computer or on a server. Some tools perform calculations locally, other on a server or both. A Desktop OLAP tool can be a cheap and simple solution or be part of a bigger package which provide tools for performing analysis on the desktop or web.
<table>
<thead>
<tr>
<th><strong>MOLAP Advantages</strong></th>
<th><strong>MOLAP Disadvantages</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized storage for multi-dimensional analysis</td>
<td>Generally limited to the analysis of summary data</td>
</tr>
<tr>
<td>Fast and consistent performance for interactive queries</td>
<td>Scalability limited by the capacity of the MDDB</td>
</tr>
<tr>
<td>Can act as departmental datamart</td>
<td>Lack of support for parallel platforms, data replication, system management software</td>
</tr>
<tr>
<td>Easy to set up and manage</td>
<td>Time for dataloading acts as limit on scalability</td>
</tr>
<tr>
<td>Strong support for advanced analytical functions</td>
<td>Multidimensional model can be inflexible to rapidly changing business needs</td>
</tr>
<tr>
<td>Support for what-if analysis requiring write access to the database</td>
<td>Requires additional data management layer</td>
</tr>
<tr>
<td>Support for sales and marketing and budgeting applications</td>
<td>A proprietary storage format (although OLE DB for OLAP is emerging as the de facto standard for data access)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ROLAP Advantages</strong></th>
<th><strong>OLAP Disadvantages</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Close integration with the data warehouse</td>
<td>Query performance is not as fast as MDDBs</td>
</tr>
<tr>
<td>Exploits existing DBMS skills</td>
<td>Requires a data warehouse</td>
</tr>
<tr>
<td>Able to analyse data along many dimensions</td>
<td>Limited range of modelling and forecasting functions</td>
</tr>
<tr>
<td>Capable of handling very large (up to terabyte range)</td>
<td>Not suitable for departmental datasets applications</td>
</tr>
<tr>
<td>Analysis is possible to the level of transactional data</td>
<td>Needs work-around for what-if type analyses</td>
</tr>
<tr>
<td>Particularly suited to product and customer management applications</td>
<td>Not suited for budgeting and financial planning applications</td>
</tr>
</tbody>
</table>

Figure 2.15: Advantages/disadvantages of ROLAP and MOLAP [12]
2.8 Analytic Visualization

Another important feature of an OLAP system is the ability to visualize multi-dimensional data. An aspect of this is to determine which representation suits a result set best. Choosing a correct representation – which can be graphical of numerical – for this data is vital to be able to correctly interpret query results. Correct representation of the multidimensional data enables a user to discover trends, constants, identify exceptions and predict future events. This requires an OLAP tool to be flexible in data visualization for users to gain insight in the patterns of the data they are analyzing.

A lot has been written on the subject of data analysis and visualization so this section will not attempt to describe all facets of this subject but merely provide knowledge of the existence of some of the most common visualization techniques for multidimensional data that all OLAP tools support.

The references through the rest of this section are good resources for further study on this subject.

2.8.1 Graphical Representation

A graphical representation can be anything visual on a two dimensional surface [20] [23]. Graphical representations can be classified in many different ways emphasizing different aspects that can be represented. OLAP tools focus on symbolic representation that is, abstract objects following a certain convention for representing information (for example lines, dots, planes which have meaning in a certain context, see figure 2.16). Figure 2.17 shows a categorization of some abstract objects for some key distinctions. Based on the characteristics of analysis data such as ordering (nominal, ordinal, cardinal) and variable types (distance, dollars for example) it is possible to choose a correct representation of the information. Figure 2.18 shows the relation between visual attributes and ordering. For example analysis data containing sales of the stores, store type and store locations of a company located in the Netherlands could be represented by a map with cylinders at the store location varying in height.

![Figure 2.16: Example of a symbolic representation useful in OLAP](image)
Figure 2.17: Key distinctions between numeric, graphic, and colored displays [21]

Figure 2.18: Ordering versus visual attributes, source: MacEachren 1995, p. 272, derived from Bertin 1967/1983 [24]

Based on the store sales (see figure 2.19). This representation makes it easier to see which type of store is profitable in what area (the locations are preserved in the representation).

Basic graphics (like pie charts and histograms) can typically only represent two or three dimensions at maximum but often the data users work with consist of more dimensions. A number of ways can make basic graphics useful with higher than three dimensions.

If a few dimensions are visualized and the other dimensions are constant this is called a subset of the analytical data. A drawback of this kind of representation is that any pattern of some kind can not be discovered in the other dimensions (like time of the year, was this a summer sale?). See figure 2.20 for a subset example.

A way to display more information is to tile visualizations with different constraints on the constant dimensions. There are also more sophisticated graphical representations. An extensive amount of research has been done on visualization
of data which have produced several tools which are very powerful in representing information. One example of this produced by Polaris [3] is shown in figure 2.21.

Figure 2.20: Line graph showing two dimensions, other dimensions constant

2.8.2 Numerical Representation

Data can be divided in two types: qualitative and quantitative. Qualitative data is nominal or ordinal, quantitative data concerns amounts or ratio and interval scales. An interval scale has meaningful when looking at difference but not ratio. These scales do not have a true zero. Ratio scales do have a true zero and therefore ratios of the values have meaning. Qualitative data can be represented by a table with values. When representing quantitative data in tables it is often very hard to discover any pattern in the
Figure 2.21: Example of a tiled scatter plot in a pivot table (subsection 2.8.3)

data and it gets increasingly difficult if not impossible when the table is bigger than a few values. Figure 2.22 taken from Thomsen’s OLAP Solutions [20] shows an example of this.

Although it does not seem to be very helpful to represent quantitative data in tabular form it is helpful when analyzing individual values, summaries or variance/ratios between single values. Special tables can represent multi dimensional data very well. Because a whole dataset cannot be visualized entirely with all possible combinations of dimensions, starting analysis by exploring the multidimensional data with these tables is a good starting point for further investigation (with or without visualization tools). These tables are called pivot tables and will be explained in the following subsection.

2.8.3 Pivot Tables

A pivot table is a table linked to a data source that enables a user to display summary data and provide the ability to move columns and rows to produce other summaries. A summary can be based on any kind of calculation like sums, averages, counts, percentages, min and max. Pivot tables make large
Figure 2.22: Store profits by store and time

Tables with quantitative data more understandable. Figure 2.23 shows a pivot table and figure 2.24 shows the same pivot table but with row and column fields exchanged (both produced by Mondrian [11]). Another advantage of pivot tables is that the data in a pivot table can be easily shown in a graphical view like a bar chart (see figure 2.25).

Figure 2.23: A pivot table in Mondrian

Figure 2.24: A pivot table in Mondrian with row and column exchanged

Pivot tables consist of two field types. Data fields contain values defined at the intersections of dimensions (known as cells) or summarized values (that are aggregations of dimensions or measures of some kind).

Row and column fields represent the dimensions and measures and determines how the summaries (and which summaries) will be displayed. Multi dimen-
ional operations including drilling up/down, slicing (moving back/forward in a dimension) and sorting are all supported by pivot tables. Figure 2.26 shows an example of a drill-down of alcoholic beverages in the products dimension.

Figure 2.25: A bar chart of the pivot tables in figure 2.23, 2.24 in Mondrian

<table>
<thead>
<tr>
<th>Product</th>
<th>Education Level</th>
<th>Unit Sales</th>
<th>Store Cost</th>
<th>Store Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alcoholic Beverages</td>
<td>Bachelors Degree</td>
<td>845</td>
<td>60,841.03</td>
<td>2,667.46</td>
</tr>
<tr>
<td></td>
<td>Graduate Degree</td>
<td>399</td>
<td>29,449.77</td>
<td>1,107.18</td>
</tr>
<tr>
<td>Beer</td>
<td>Bachelors Degree</td>
<td>145</td>
<td>3,610.68</td>
<td>359.70</td>
</tr>
<tr>
<td></td>
<td>Graduate Degree</td>
<td>99</td>
<td>317.87</td>
<td>230.71</td>
</tr>
<tr>
<td>Wine</td>
<td>Bachelors Degree</td>
<td>700</td>
<td>57,230.35</td>
<td>2,307.76</td>
</tr>
<tr>
<td></td>
<td>Graduate Degree</td>
<td>300</td>
<td>29,131.90</td>
<td>876.47</td>
</tr>
<tr>
<td>Beverages</td>
<td>Bachelors Degree</td>
<td>2,018</td>
<td>119,816.99</td>
<td>6,619.61</td>
</tr>
<tr>
<td></td>
<td>Graduate Degree</td>
<td>1,040</td>
<td>54,418.68</td>
<td>3,254.69</td>
</tr>
</tbody>
</table>

Figure 2.26: Drill down into alcoholic beverages of the pivot table in figure 2.23
Chapter 3

OLAP Products

3.1 Introduction

The introduction of this thesis suggested that OLAP is a tool for analyzing and performing calculations on multidimensional data. Section 2.2 of chapter 2 introduced the most widely accepted characterization of an OLAP system. Considering the concepts in the previous chapter and the characterization of OLAP, how does this all relate to OLAP products? In other words, what can we expect from an OLAP product?

Most OLAP products focus on large companies or enterprise that generate and collect large amounts of operational data that is usually stored in a RDBMS. A few examples of operational data are sales and costs of products or ATM transactions of a bank (which contain information about time of the transaction, amount of money collected, date, and so on). Business analysts want to analyse this operational data for decision oriented activities.

RDBMS systems are optimized for storage and retrieval of operational data. This is also called OLTP (On-Line Transactional Processing). Operational activities typically have the following characteristics:

- Frequent access of data
- Small amount of data accessed per query
- Data access is well-defined and predictable
- Operates on raw data
- Often requires the most current data

OLAP and Decision Support Systems (DSS)\(^1\) in general need a high level view on data. These systems are designed to answer questions like ‘Has the company

\(^1\)This includes Data mining: information extraction activity with the goal to discover hidden facts in data stored in a database system
made more profit this year compared to a year ago?’ or ‘What is the income of the average customer that bought our product?’ by organizing consolidated source data in dimensions and measures. This of course means that an OLAP system needs to meet other requirements and therefore has other characteristics:

- Unpredictable data access (Needs to support the line of thought of a business analyst)
- Large amounts of data accessed per query (Sales this year compared to last year)
- Often operates on derived data
- Sporadic access of data
- Needs historical as well as current data (Displaying sales in a graph over five years to discover a trend)
- Complex derivations

Not only do OLAP systems need to deal with the problems mentioned in the previous chapter, these systems also need to provide an intuitive way of interacting with multidimensional data, fast response time and should offer many different ways of visualizing data.

The client-side of OLAP tools usually consist of a query language and reporting tool. A lot of tools offer an interactive environment (which can be web-based) where the multidimensional data can be analyzed without having to write queries if the solution offers a query language at all. This minimizes the learning curve of a tool and makes it easier to use.

Because spreadsheets have been widely used for years and people are used to a spreadsheet environment most products offer (or are only available as) a spreadsheet plugin. This way users can still use their familiar environment but then also have the tools to analyse multidimensional data.

The following sections introduce two OLAP products, a commercial product which has gained a big market share in the OLAP business and the only serious open source OLAP product to give an idea what to expect from a product. Appendix C contains a list of the most popular OLAP products available with references to their web-site. For a comparison of the performance between OLAP products see The OLAP Survey of N. Pendse at the web-site of survey.com [18].

### 3.2 Microsoft SQL Server – Analysis Services

#### 3.2.1 Overview

Microsoft’s entry in the OLAP market began with the acquisition of Plato, the OLAP technology of Panorama Software Systems market leader of OLAP in Israel (Cognos later bought client and web connectivity tools from the same company to provide an interface to Microsoft’s OLAP server released as NovaView,
The Analysis Services architecture supports ROLAP, MOLAP and HOLAP. Microsoft extending their OLE (Object Linking and Embedding) DB for OLAP, a set of COM (Component Object Model) interfaces for uniform data access to heterogeneous information resources which is used by Analysis Services to access different datasources through its OLE DB/ODBC component and interact with client tools through OLE DB/ActiveX controls. OLDE DB also provides the ability to create additional database services.

In a nutshell, Microsoft’s OLAP product is an example of the 80/20 principle. Eighty percent of the users will be happy with a simple and easy to use OLAP solution. The remaining twenty percent can be taken by adding features later on.

### 3.2.2 Features

The following list will show a summary of the most important features of Analysis Services:

- All hierarchy types are supported.
- As explained in the overview with OLE DB users can develop applications that can interact with Analysis Services to offer new datasources, build web-based clients or integrate OLAP functionality in their tools. Applications can query OLAP databases using MDX (Multi Dimensional eXpression language).
- Cube partitioning to increase performance cubes can be partitioned for parallel access or to aggregate only parts of a cube (most frequently used aggregations for example).
- Flexible data model which supports ROLAP, MOLAP and HOLAP.
- Incremental updates of OLAP data without the need to reload and reaggregate all data.
- Supports Clustering of servers.
- Wizards for any important task:
  - **Calculated cells wizard** allowed cells values to be calculated according to a formula written in MDX.
  - **Cube editor/builder wizard** to help create a logical OLAP model.
**Dimension editor/builder wizard** to design and edit dimensions, supports all kinds of hierarchies.

**Storage design wizard** to aid in determining what kind of storage method to use, including tools to optimize the amount of aggregation versus response-time and memory usage.

**Usage analysis wizards** to analyse how a cube is used, and tools to modify aggregations to the user’s need.

**Virtual cube wizard** for easy creation of virtual hypercubes.

- Write enabled cubes, data can be written back to the datasource.

### 3.2.3 Architecture

![Architecture Diagram](image)

Figure 3.1: The architecture of Analysis Services

Figure 3.1 taken from the MSDN site about Analysis Services shows the server architecture of Analysis Services.
The heart of Analysis Services is the OLAP engine. The OLAP engine has a storage layer which is responsible for storage and retrieval of data from different datasources and also for storage of metadata (information about the logical OLAP models which define cubes, dimensions and so on).

Analysis Services can communicate with client tools in two ways, the first is through the Manager interfaces and the second is through the PivotTable Service.

The PivotTable service is an OLE DB provider that supports the OLAP extensions of OLE DB but does also support non-OLAP OLE DB. It functions as a client to Analysis services but also has caching capability and can store data locally for offline analysis. The PivotTable Service can also create local cubes or communicate with Analysis to modify and update cubes.

The Object Model or Decision Support Objects (DSO) provide COM interfaces to create applications that can define, modify and manage cubes. It functions as the glue between applications and the Analysis Services and can be used to automate system maintenance or to extend the functionality of the Managers.

A layer based on DSO provides the managers for the Server component and OLAP modeling components of the Analysis Services. The first is the Enterprise Manager which provides a user interface with the server. It does also include the Data Transformation Services (DTS) component and provides graphical user interfaces and COM interfaces to import, export and transform data for tasks as data consolidation, archiving or analysis.

The second management tool is the Analysis Manager. This component is used to create, define and manage cubes, optimize query performance, browse data sources, dimensions, security roles and other objects. It also has a COM interface and a ‘Add-in’ component.

Applications can communicate with Analysis Services in four ways using COM interfaces of the Analysis Add-in Manager, Object Model, PivotTable Service or a combination of these components.

The Analysis Add-in Manager provides components to add functionality to the Analysis Manager or to use the functionality of the Analysis Manager in custom applications.

All information about data (metadata: definition of the cubes, dimensions, OLAP model and so on) is stored in the Metadata Repository. This is a relational database and the stored information can be accessed with the Metadata Services COM interface.

Microsoft Management Console (MMC) is a common presentation service for management applications. It contains amongst other management applications the Enterprise Manager and Analysis Manager.

Because the architecture of Analysis Services is very open developers can easily build or extend components with the available COM interfaces and in any language that supports the COM architecture (from VB to C#).
3.2.4 Security

Analysis Services has different stages of security. The first stage is to authenticate on the system. The only option available is Windows Authentication. This means that to be able to use Analysis Services there should exist a windows user account for that user. Although users can also log in using http with IIS (Internet Information Services, Microsoft’s web-server) users eventually need to be mapped to a windows account.

When a user has a windows account and is in the appropriate group with the correct permissions, it can be assigned security. A role defines a set permissions/actions that a specific user can have. Managers at a company for example need permission to analyse and view business data, but secretaries are only allowed to enter or lookup administrative data in a part of the system. There are three standard security roles:

- **database role** Defines read/write access on specific cubes.
- **cube role** applies to a single cube and derived from a database role. A cube role can define cell permissions.
- **mining role** This role is essentially the same as the database role but applies to a mining model (which is out of the scope of this thesis).

Analysis Services support cell- dimension-, application-level and even MDX security. The latter defining the access based on a MDX query. Users can be a member of multiple roles.

Considering the fact that security is based on windows authentication does not affect the systems ability to communicate with other platforms. Microsoft has developed a standard in cooperation with other companies for accessing and exporting OLAP data.

3.2.5 Standards

Microsoft provides XML for Analysis [7] for short. The XML/A specification was originally sponsored by Microsoft and Hyperion, now twenty seven companies are working on this standard. It defines a set of XML messaging interfaces for exchanging OLAP data over the internet using SOAP [2] (Simple Object Access Protocol), an already widely accepted standard. XML/A is becoming the standard for OLAP communication.

A lot of vendors have also implemented the OLAP OLE DB API.

3.3 Mondrian

3.3.1 Overview

Mondrian is an open source effort to implement a ROLAP server in the Java programming language. The project was founded by J. Hyde in september 2001.
The 1.0 version of Mondrian was released at the 17th August 2003 and claims to be the first production quality Open Source OLAP Server [11]. The Mondrian OLAP Server basically consists of three parts: The OLAP Engine, a RDBMS containing the analysis data and a reporting tool for performing analysis on OLAP data (see figure 3.2). JPivot – an Open Source project founded by A. Voss – is a JSP [15] custom tag library to render OLAP data cubes and let users perform typical OLAP navigations [25] which is used by Mondrian as the reporting tool. Mondrian can run on the most popular databases including Oracle, MySQL, Microsoft SQL Server and others. Because the OLAP server is entirely written in Java it is easy to use and integrate in existing Java applications.

![Figure 3.2: Architecture of Mondrian](image)

### 3.3.2 Architecture

The following subsections discuss the OLAP engine of the Mondrian system, how to define your multidimensional data and map this to data stored in a RDBMS, security, standards implemented by Mondrian, the included reporting tool and an opinion about the system.

#### OLAP Engine Layers

The OLAP engine of Mondrian consists of two parts: the calculation layer and the aggregation layer. The responsibility of the calculation layer is to process queries (parsing, validating, executing) and send cell requests to the aggregation layer to retrieve and process the data. The calculation layer sends the request in such order that first the axis are computed and later the cells in between. The calculation layer can also transform queries in order to speed up queries that are similar in structure instead of creating statements from scratch for each query.
The aggregation layer retrieves data from the underlying relational database and is able to cache the retrieved cells. If a database supports optimizations for data retrieval such as materialized views (a database object that contains results of a query) this layer can be enabled to use these optimizations. When a request of the calculation layer is only partially or not at all in the cache the aggregation layer send the request to the RDBMS. The aggregation layer is also responsible for calculating aggregations of cells (for example a level summary also called a roll-up).

Because Mondrian is a ROLAP tool and the aggregation layer uses an adaptive caching scheme it does not suffer much from database explosion. Only requested aggregated data is kept in memory and if the aggregated data grows bigger than the memory size cells can be dumped. These cells have to be recalculated when requested again however. For this approach to be efficient the caching mechanism needs to make the right choices when dumping cells based on frequency of use, calculation time of a cell, cell size and so on.

### Metadata Definition

In order to know how to build queries and make use of database optimizations (like materialized views) the aggregation layer needs to know how the data is stored. Metadata contains the description of the logical OLAP model and how it is mapped to the RDBMS (physical model). In Mondrian this metadata has to be defined according to a xml schema. In other words, the xml schema contains among other things cubes, hierarchies and measures that define the logical model and how these things are mapped to the tables in the RDBMS, the physical model.

Figure 3.3 shows a sample Mondrian schema. A Cube element defines the measures and dimensions that are part of a certain OLAP cube. Table elements define the mapping between a dimension or measure to tables or views in the RDBMS containing the data. Dimension and Measure elements define the dimensions and measures of the OLAP model.

Mondrian supports parent-child, leveled and multiple hierarchies. It does also support joining cubes with shared dimensions (virtual hypercubes). To use this feature the shared dimensions of the cubes that can be joined have to be defined in the xml schema first. Please refer to the Mondrian metadata dtd for an explanation on all elements.

### 3.3.3 Security

In a multi-user environment security becomes an issue when not all users are allowed to view or analyse all available data. Therefore multi-user OLAP tools provide some way of acces-control. Mondrian implements role-based access-control.

The access-control profile of Mondrian can be defined in the xml schema which defines the OLAP model and relational mapping or can be controlled programmatically. Access-control can be applied to schemas (a complete defined model),
Figure 3.3: Sample xml schema in Mondrian part of the Mondrian documentation [11]
cubes, hierarchies and members of hierarchies or parts of cubes, hierarchies and members. With these features the access-control of Mondrian is very versatile.

### 3.3.4 Standards

Before Mondrian came to life there were already a lot of OLAP vendors with their proprietary standards. Although there is not yet a widely accepted standard in the OLAP world for communications between OLAP servers, query languages or export of OLAP data, some of the standards used by Microsoft (and its collaboration with other companies) have been adopted or are supported by other companies.

Instead of inventing a new query language Mondrian adopted the popular Microsoft query language MDX \[5\] (Multi Dimensional Expression language) of Microsoft’s Analysis Services. See appendix B for a short description of this query language.

Mondrian does also implement XML/A.

The primary API (Application Programming Interface) of Mondrian is Mondrian proprietary. A collaboration of companies (amongst other companies Sun, Microsoft, Hyperion, IBM, Oracle and SAS) which have heaviily invested in J2EE have been working on a standard API for Java J2EE which addresses the programming needs developers have when working with OLAP. This API is called JOLAP (Java OLAP Interface) and at the time of this writing the final version of the API specification has not yet been released. Mondrian is moving towards this standard and has already implemented part of the final proposed draft specification \[14\].

### 3.3.5 Reporting Tool

Because the architecture of Mondrian is modular this enables anyone to write a reporting tool that can visualize the Mondrian OLAP data in Java or any other language using the XML interface of Mondrian. For users that do not want to write such a tool Mondrian is provided with JPivot, an open source web-based reporting tool for visualizing OLAP data. This section takes a look at some of the features of JPivot.

Figure 3.4 shows a typical JPivot view in drill position (see the fifth item of the following list). mode. A JPivot view contains OLAP data configuration options as well as view options. Providing a full explanation of the available options is out of the scope of this thesis. See the JPivot website for detailed information about visualizing OLAP data with this tool.

The following list explain a few typical OLAP operations that JPivot support:

- **MDX**: Shows the MDX editor to create views on the OLAP data cube.
- **Suppresses empty rows/column**: Suppresses empty rows/column.
- **Swaps the row/column fields**: Swaps the row/column fields.
Navigates through a member, showing the base data (joined dimension and fact tables).

Navigates through the hierarchy.

Replaces a hierarchy with the members.

Shows base data (see member navigation).

Creates a chart with the current pivot table as base data (see figure 2.25).

3.4 Summary

Microsoft Analysis Service is a very user friendly solution offering the most important tools for OLAP. It has a query language, visualization tools, ROLAP, MOLAP and HOLAP support, good exporting capabilities and an open architecture which makes building both web-based and normal OLAP enhanced applications very easy. The COM interfaces of Analysis Services gives full access to the server. Not only is Analysis Services a nice OLAP solution, compared to other solutions it is a very cheap one. This makes it available to many small companies with a minimum of costs. A downside of Analysis Services is that it is only restricted to one platform.

The main advantage of Mondrian is that it is free and it is the only serious open source OLAP effort known (except for a research project called Lemur, see [8]). But it has not fully matured into a product yet. There are no easy tools for installing the system or building an OLAP model (although a graphical
workbench is being developed for this task) for example. Mondrian can operate
on a normalized database (which is used for OLTP), but this setting will have
a negative impact on the performance. It is necessary to manually build a
warehouse based on the operational data and let Mondrian operate on this.
The project does also suffer from poor documentation and the performance of
the OLAP engine cannot match with the performance of any of the common
commercial solutions (like TM1 or Microsoft Analysis Services). Most commer-
cial products have been developed in a significantly longer time though.
Nevertheless Mondrian is very suitable to be embedded in Java applications
that need OLAP functionality or to serve as the OLAP engine for a custom
OLAP solution. Another advantage is that the OLAP engine and reporting tool
of Mondrian are platform independent and could be distributed over different
machines.
The project has a small dedicated group of developers working on improving
Mondrian and has a large interested audience as the forums on the Mondrian
project site proves. It is also the only open source project implementing the
upcoming JOLAP standard. Therefore it has a lot potential for the future.
Chapter 4

Conclusion

The ability to take advantage of all the available information is a critical component for the success of an enterprise. As the available business information to an enterprise is constantly growing it is getting increasingly difficult to manage. To improve business efficiency, performance and to support decision making businesses analyze their operational data. Big enterprises have been investing in Business Intelligence (BI)\(^1\) systems for years resulting in technologies as OLAP and Data Mining (the latter having roots in artificial intelligence, statistics and machine learning) that consists of a number of technology layers as this thesis has shown.

It is not an easy task to define the exact terms that characterize an OLAP system. Codd’s rules proved to be too restrictive. Pendse and Creeth developed the FASMI characterization to provide a simpler model that characterize OLAP systems. This has been used as the basis of this thesis to explain all facets of an OLAP system.

BI solutions that companies like Hyperion, Applix and Microsoft offer have proven to be very successful in improving efficiency and reducing costs in enterprises as a lot of case studies show. Because the most OLAP solutions are very expensive these solutions are often not in reach for small companies who can not afford the high costs. Although there are a few efforts to create an open source OLAP server they can not compete with the commercial products. Microsoft entered the OLAP market with the aim to build an affordable product projected at the lower end of the business. They succeeded in their approach and this resulted in an OLAP product which is affordable even for very small companies.

Not only has OLAP proven to be a very useful technology to BI, it has also been applied to other kinds of problems like finding nodes for location-based services [9]. The story of OLAP is a story of success and it is gaining territory in other area’s where we are presented with large amounts of multidimensional information to process.

\(^1\)A term used to indicate analysis on operational business data to aid decision support
Chapter 5

Future Work

Long before the term OLAP was coined (1993) multidimensional systems have been a field of research. Since the 1960s researches have been trying to model business operations using multidimensional matrices and operations on them. Current OLAP systems can now handle data sizes ranging from small databases to very specialized systems that can process terrabytes of data.

Because multidimensional information processing was a problem at big enterprises which needed a way to analyse their business data to be more competitive a number of commercial products have been developed which have succeeded in providing a good environment for data analysis.

The only open source effort of production quality is Mondrian which still has a long way to go before it can compete with the commercial products. A subject for future work on Mondrian could be to study the quality of the Mondrian OLAP server and bring into map the virtues and the points of improvement because there does not yet exist a complete assessment of this project.

Short term points of improvement could be to optimize the efficiency of the aggregate layer or to build a tool that can automatically generate a warehouse based on the operational database and requirements given by the user.

An interesting topic that has great importance in OLAP is how to visualize datasets to retain as much information as possible in the visual representation. J. von Engelhardt provides a syntactic framework for graphic visualization in his PhD thesis The language of Graphics. Future research could be to automatically generate and explore visual representations based on OLAP data using his work.
Appendix A

Codd’s 12 rules

This appendix contains a summary of Codd’s twelve rules to which OLAP systems have to conform:

1. Multidimensional Conceptual View
   Because the analyst view on the enterprise is multidimensional in nature an OLAP tool that facilitates data analysis should be multidimensional in nature too.

2. Transparency
   It should be transparent to the analyst where the data came from and any form of the tool be it a spreadsheet add-in or stand-alone application should get full value from an OLAP engine. This means that an OLAP tool should be able to use data from heterogeneous data sources and that data could be accessed in all possible ways (like direct, in the form of a query language or indirect with visualization tools).

3. Accessibility
   An OLAP tool should have its own logical schema to access heterogeneous data and present it to the user in a single, coherent and consistent manner. This basically means that an OLAP engine is a system that sits between data sources and reporting processing, converting or retrieving data for presentation.

4. Consistent Reporting Performance
   An OLAP system should have constant performance while performing analysis. When the database size increases this should not result in degraded performance of the tool.

5. Client-Server Architecture
   Because analysis will be mostly done on workstations OLAP tools should able to operate in a client-server environment. The server must be capable of consolidating data from different data sources and handling a number of clients.
Various client should be able to attach to the server with a minimum of program- ming and effort.

6. Generic Dimensionality
All dimensions should have the same operational capabilities and be equivalent in structure. The basic datastructure, formulae and reporting formats should not be biased toward any dimension.

7. Dynamic Sparse Matrix Handling
OLAP tools have to provide methods to change physical data access and indexing. Adopting to various data distributions (one of which is sparseness of data) for maximum memory efficiency and performance should be possible. Also does it have to handle missing or inapplicable values in a correct manner.

8. Multi-User Support
Multiple users have to be able to work on the same system simultaneously. Therefore an OLAP system have to provide concurrent data access (retrieval and update), integrity and security.

9. Unrestricted Cross-dimensional Operations
All dimensions should support all forms of calculations, not only just the measures. Relation models are very weak in this area, but multidimensional products mostly are strong at this.

10. Intuitive Data Manipulation
Operations like drilling down across columns or rows, drill through and other manipulations in the hierarchy of dimensions should be possible to do as actions directly on the cells of the analytical model and not with the use of any menu's or other trips across the user interface.

11. Flexible Reporting
It should be possible to represent data in other that tabular form. An OLAP application has to provide many possible ways to visualize the analytical data.

12. Unlimited Dimensions and Aggregation Levels
Any serious OLAP tool should at least support fifteen to twenty dimensions in an analytical model. Also should it be possible to define any number of aggregation or consolidation levels in the dimensions.
Appendix B

MDX Example

This appendix introduces MDX, a highly functional expression syntax for creating querying and performing calculations on multidimensional data in Microsoft SQL Server. It summarizes some of the key points of the MDX language. For in-depth documentation on MDX see the OLE DB Programmer’s Reference [5].

One of the simplest forms of an MDX expression is:

```
SELECT axis specification ON COLUMNS,
  axis specification ON ROWS
FROM cube_name WHERE slicer_specification
```

An axis specification can be seen as a selection on a dimension. In this example COLUMNS and ROWS are dimensions indicating a select on two dimensions. For more dimensions the naming convention AXIS(index) can be used indicating the dimensions by index in a cube. The slicer_specification determines which dimensions of a cube will be constant (so a subset or slice of the cube will be viewed).

The next query shows how to specify members of dimensions and measures:

```
SELECT Measures.MEMBERS ON COLUMNS,
  [Store].MEMBERS ON ROWS
FROM [Sales]
```

This query takes all members from the Measures dimension (The special dimension containing the facts) for every store Store dimension and creates a summary at every defined summary level (a defined aggregation, say All Stores) from the Sales cube. It displays all the members of the measure for the store hierarchy.

The following query shows an enumeration of members in the Store hierarchy selecting the children of these members:

```
SELECT Measures.MEMBERS ON COLUMNS,
```
{[Store].[Store State].[CA].CHILDREN,
    [Store].[Store State].[WA].CHILDREN} ON ROWS
FROM [Sales]

This queries the measures for the children of the states California and Washington summarizing the stores.

Drilling up/down a hierarchy can be done with the function:

DESCENDANTS(member, level [, flags])

This function takes a member and a level to descent to. Flags can be: BEFORE, AFTER, BEFORE_AND_AFTER to indicate the levels before, after or before and after the level specification.

DECENDANTS([Store].[Store State].[CA], [Store City], AFTER)

This will summarize at state level to store name.

Slicer specifications work similar to member specification so the following query would return the sales average for 1997.

WHERE (Measures.[Sales Average], [Time].[Year].[1997])

For hierarchical navigation there are the following constructs: PREVEMBER, NEXTMEMBER, CURRENTMEMBER, PARENT. There are also other constructs like FIRSTCHILD and LASTCHILD.

The last example is a query that calculates the percentage of a product brand as a percentage of the sales of that product within its product category:

WITH MEMBER MEASURE.S.PercentageSales AS
    '([Product].CURRENTMEMBER, Measures.[Unit Sales]) /
     ([Product].CURRENTMEMBER.PARENT, Measures.[Unit Sales])',
    FORMAT_STRING = '#.00%'
SELECT {MEASURES.[Unit Sales], MEASURE.S.PercentageSales} ON COLUMNS,
    [Product].[Brand Name].MEMBERS ON ROWS
FROM [Sales]

These examples were taken from the MDX tutorial which is part of the OLE DB documentation available at the Microsoft SQL Server home page [5].
Appendix C

Product List

This appendix contains a summary of available products on the market today. It is far from complete but providing a complete list is out of the scope of this thesis. However, this list helps in evaluating OLAP products by providing a starting point for evaluation.

Resources that were used for determining which products to mention are *The OLAP Report*, *Ovum Evaluates OLAP* and internet resources.

COMMERCIAL TOOLS:

**Applix TM1** [http://www.applix.com](http://www.applix.com) According to *The OLAP survey 3* (2003) Applix TM1 has the best performing analytics engine of all tested products. This is quite an achievement as *The OLAP survey 3* is the largest and most detailed survey of OLAP products available (1047 OLAP sites with 42 different products). Applix has a range of OLAP products from a small web-based OLAP server to a complete set of servers and applications for large organizations. The core of TM1 is the RAM based MOLAP engine which is extremely efficient in storing operational data. Because it is RAM based it can provide high performance on real-time calculations.

**Arbor Essbase** Merged into Hyperion Software.


**Cognos PowerPlay** [http://www.cognos.com](http://www.cognos.com) This tool focusses on large amounts of data (billion rows of data with two million categories for example). Cognos Powerplay is a multi-tier server architecture that can run on multiple platforms. It supports multiple vendors like Microsoft SQL Server, Essbase, IBM OLAP, and so on. It also has both ROLAP and MOLAP capability.

**Crystal Decisions Holos** The first HOLAP server on the market that could access relational and multidimensional data simultaneously.
Express The first tool aimed at marketing applications. Owned by Oracle.

Hyperion Essbase http://www.hyperion.com Essbase is a high-end multi-platform MOLAP server which offers a broad range of tools for both developers and end-users.

IBM DB2 OLAP http://www-306.ibm.com/software/data/db2/db2olap/ DB2 is IBM’s RDBMS. To provide OLAP IBM uses Essbase as it’s OLAP engine. This ROLAP product load and calculation processes are slower than the original product because of the Essbase-DB2 mapping.

Microsoft Analysis Services http://www.microsoft.com/sql/evaluation/bi/default.asp This OLAP product is part of the Microsoft’s SQL server product and supports ROLAP, HOLAP and MOLAP. It has good exporting capabilities (XML/A) and can be easily integrated into web-based solutions with the use of ASP components (and more recently ASP.NET). With Microsoft’s COM framework developers can quickly build applications that use the OLAP server using Analysis Server’s OLE DB. OLE DB provides a set of COM interfaces that provide applications with a uniform way of accessing data (…) [5].

Microstrategy 7i OLAP Services http://www.microstrategy.com The first ROLAP server without a multidimensional engine. The modern version (7i) has a Hybrid architecture.

Oracle 9i OLAP Release 2 http://www.oracle.com/olap/ Oracle offers an OLAP solution that is embedded in their database product. It uses Express as it’s OLAP engine and supports both ROLAP and MOLAP. They claim to be 75 times faster according to the APB-1 benchmark of the OLAP Council. (This can not be taken too seriously as the OLAP council is no longer in existence and it’s successor the Analytical Solutions Forum has also disappeared due to some failed attempts to provide standard for OLAP). The ABP-1 benchmark which has not been used for years, with the product running on better hardware and a more mature operating system (Sun Solaris compared to the scalability problems of Windows NT) than the other products, Oracle compared it’s results with a four year old benchmark which is not representative for the current status of other products). Nevertheless Express has survived for more than thirty years and still holds a considerable market segment.

OPEN SOURCE TOOLS:

Howler http://www.daniel-lemire.com/OLAP/ A web-based OLAP application based on the open source HOLAP C++ library Lemur (http://savannah.nongnu.org/projects/lemur/). Lemur is a research project.

Mondrian http://mondrian.sf.net ROLAP Server project founded by J. Hyde. It is written in the Java programming language and includes a basic reporting tool JPivot.
Bibliography


Index

Access-control
  role-based, 29, 31
Aggregate, 8, 15

Base table, see Fact table
BI, 36

Cell, 22
COM, 26
Cubes, 12

Data
  inapplicable, 14
  missing, 14
  qualitative, 20
  quantitative, 20
  sparsity, 13

Data mining, 24, 36
Data visualization
  sophisticated, 19
  subset, 19
  tile, 19

Data warehouse, 14
Dimension, 5
DOLAP, 15
DSO, 28
DSS, 24
DTS, 28

Fact table, 8
FASMI, 4

Field
  column, 22
  data, 22
  row, 22

Hierarchy, 5
  deviant, 6
  leveled, 6, 9, 31
  multiple, 6, 10, 31
  parent-child, 9, 31
  ragged, 6, 9

HOLAP, 15
Hypercubes, see Cubes

Logical model, 31

Materialized view, 31
MDDB, 16
MDX, 33
Measure, 5
Member, 12
Metadata, 31
MMC, 28
MOLAP, 15
Mondrian, 29
  aggregation layer, 30
  calculation layer, 30
Multi domain, 12

OLE DB, 26
OLTP, 24

Ordering
  cardinal, 7, 18
  nominal, 7, 18
  ordinal, 7, 18

Physical model, 31
Pivot table, 21

RDBMS, 15
Roll-up, 31
Single domain, 12
Slice, 23
Snowflake schema, see Star schema
Star schema, 8

The OLAP Survey, 42

Variable, see Measure
Visual attributes, 18
Visualization, 18

XML/A, 33