Online Analytical Processing (OLAP) for Decision Support

Nenad Jukic[1], Boris Jukic[2], and Mary Malliaris[1]
[1]School of Business Administration, Loyola University Chicago, Chicago, IL, USA
[2]School of Business, Clarkson University, Potsdam, NY, USA

Online Analytical Processing (OLAP) refers to the general activity of querying and presenting text and number data from data warehouses and/or data marts for analytical purposes. This chapter gives an overview of OLAP and explains how it is used for decision support. Before the specific OLAP functions and platforms are presented, the connection between the OLAP systems and analytical data repositories is covered. Then, an overview of functionalities that are common for all OLAP tools is presented.
Keywords: OLAP; Dimensional Model, Data Warehouse; Data Mart

1. Introduction

The purpose of this chapter is to give an overview of OLAP and explain how it is used for decision support. OLAP is an acronym that stands for Online Analytical Processing. It is important to note that the term OLAP followed the development of the standard database concept OLTP – Online Transactional Processing.

OLTP refers to the general activity of updating, querying and presenting text and number data from databases for operational purposes. In other words, OLTP encompasses all the everyday transactions done on the operational database systems, such as, for example, a transaction reflecting a withdrawal from a checking account or a transaction creating an airline reservation. In fact an often-used technical term for an operational database is the "OLTP system".

OLAP refers to the general activity of querying and presenting text and number data from data warehouses and/or data marts for analytical purposes. While OLTP is used in conjunction with traditional databases for operational (day-to-day) purposes, OLAP works (as is described in the next section) with the data from data warehouses and data marts. Another difference between OLAP and OLTP is that the process of OLTP includes "updating, querying and presenting" whereas OLAP includes only "querying and presenting". While OLTP systems routinely perform transactions that update, modify and delete data from databases, OLAP tools are "read only". They are used exclusively for the retrieval of data (from analytical repositories) to be used in the decision making process. Users of OLAP tools can quickly read and

interpret data that is gathered and structured specifically for analysis, and subsequently make fact-based decisions.

Both OLTP and OLAP pre-date the Internet era. The expression "Online", used by both of these terms, is not associated with the Internet or the World Wide Web. Instead, the term "Online" in these two acronyms simply refers to a type of computer processing in which the computer responds immediately (or at least very quickly) to user requests. In today's world, we are accustomed to the fact that the computers perform processing, updating and retrieving of data instantaneously. However, at the time the term OLTP was created, many of the computers still used devices such as magnetic tapes and punch-cards readers. The expression "Online" was used to underscore the immediacy of the results, where databases systems used a direct access type of storage (such as a hard drive) instead of a sequential access storage device (such as a magnetic tape).

Before the specific OLAP functions and platforms are presented, it is important to understand the connection between the OLAP systems and the data repositories designed specifically for data analysis (i.e. data warehouses and data marts.) The next section gives a brief overview of data warehouses and data marts as they pertain to OLAP. Following the overview, the basic OLAP functionalities common across most OLAP applications are covered. The database models used by OLAP are then discussed. Next, well-known variations on the OLAP model are covered. Finally, a summary concludes the chapter by describing the overall value of OLAP.

2. Background: Data Warehouses and Data Marts

A typical organization maintains and utilizes a number of operational data sources. These operational data sources include the databases and other data repositories which are used to support the organization's day-to-day operations. A data warehouse is created within an organization as a separate data store whose primary purpose is data analysis for the support of management's decision making processes (Inmon, 2002). Often, the same fact can have both operational and analytical purposes. For example, data

describing that customer X bought product Y in store Z can be stored in an operational data store for business-process support purposes, such as inventory monitoring or financial transaction record keeping. That same fact can also be stored in a data warehouse where, combined with vast numbers of similar facts accumulated over a time period, it is used to analyze important trends, such as sales patterns or customer behavior.

Why store any fact in two places? There are two main reasons that necessitate the creation of a data warehouses as a *separate* analytical data store. The first reason is the performance (speed) of queries. Operational queries are mostly short and fast, while analytical queries are complex and consume significant amount of time. The performance of operational queries can be severely diminished if they have to compete for computing resources with analytical queries. The second reason lies in the fact that, even if performance is not an issue, it is often impossible to structure a database which can be used (queried) in a straightforward manner for both operational and analytical purposes. Therefore, a data warehouse is created as a separate data store, designed for accommodating analytical queries. A typical data warehouse periodically retrieves selected analytically-useful data from the operational data sources. For any data warehouse, the infrastructure that facilitates the retrieval of the data from the operational databases into the data warehouses is known as ETL, which stands for Extraction, Transformation and Load. Figure 1 illustrates this process.

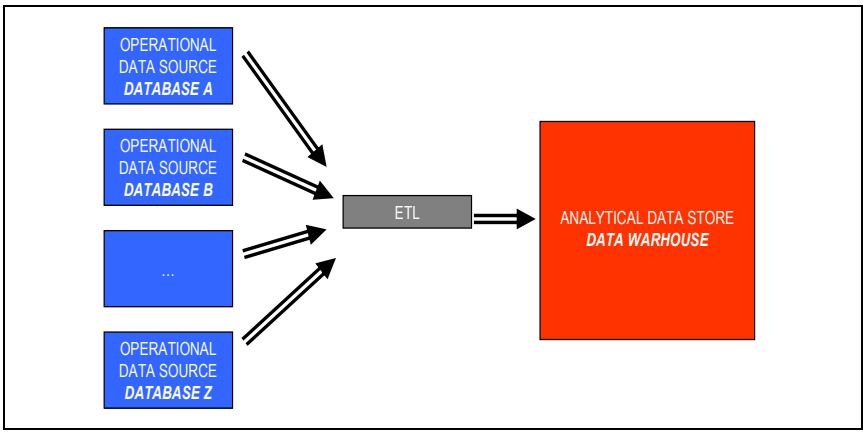


Figure 1: Data Warehouse - a Separate Analytical Repository

A data mart is a data store based on the same principles as a data warehouse, but with a more limited scope. Whereas a data warehouse combines data from operational databases across an entire enterprise, a data mart is usually smaller and focuses on a particular department or subject. Dimensional modeling (Kimball, 1998) is a principal data mart modeling technique (which can also be used as a data warehouse modeling technique). It uses two types of tables: facts and dimensions. A fact table contains one or more measures (usually numerical) of a subject that is being modeled for analysis. Dimension tables contain various descriptive attributes (usually textual) that are related to the subject depicted by the fact table. The intent of the dimensional model is to represent relevant questions whose answers enable appropriate decision-making in a specific business area (Chenoweth, 2003).

The following figures (Figures 2a, 2b, 2c, and 2d) illustrate an example where dimensional modeling is used to design a data mart that retrieves data from two operational relational databases. This example will demonstrate the important characteristics of dimensional modeling, even though, due to the space limitations, the number of tables and the amount of data is very small when compared to a real world scenario. Figure 2a shows two separate operational databases, database *A* and *B*, for a retail business. Figure 2b shows sample values of the data stored in databases *A* and *B*. The operational database *A* stores information about sales transactions. In addition to transaction identifier and date, each sale transaction records which products were sold to which customer and at which store. Operational database *B* stores information about customers' demographic and credit rating data. As most people that have been involved in maintenance or even just simple use of corporate databases can testify, multiple separate non-integrated databases are often present in real-world businesses and organizations. The reasons for the existence of multiple non-integrated databases within the same company can vary from historical (e.g. a merger of two companies with distinct database systems) to organizational (e.g. decentralized departmental structure of a business) and technical (e.g. various software and hardware platforms present to support various processes). In this example, let us assume that the database A is kept by the retail business' sales department in order to record and process sales data, whereas database B is populated with the data

acquired from an outside market-research company in order to support the marketing initiatives. Therefore, the customer IDs in database B match those in database A.

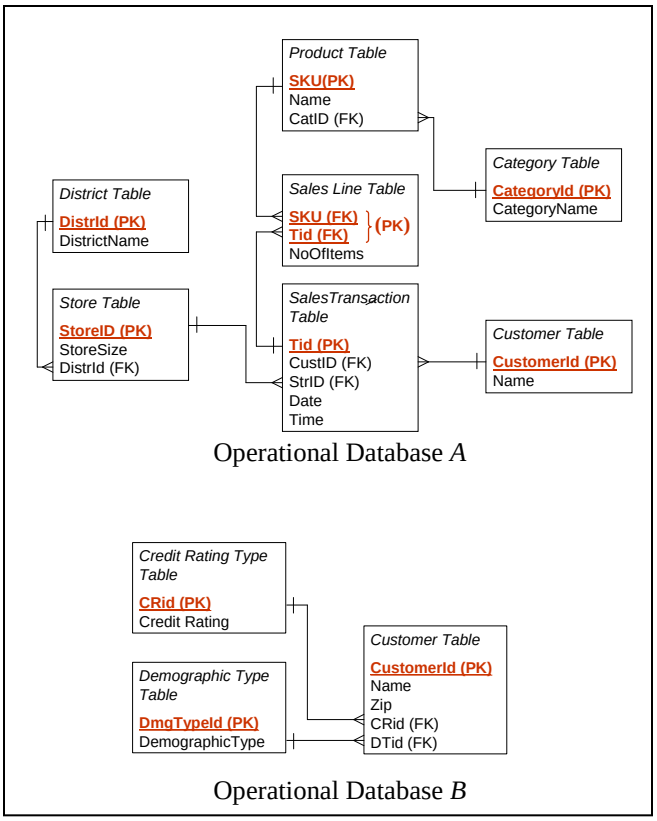
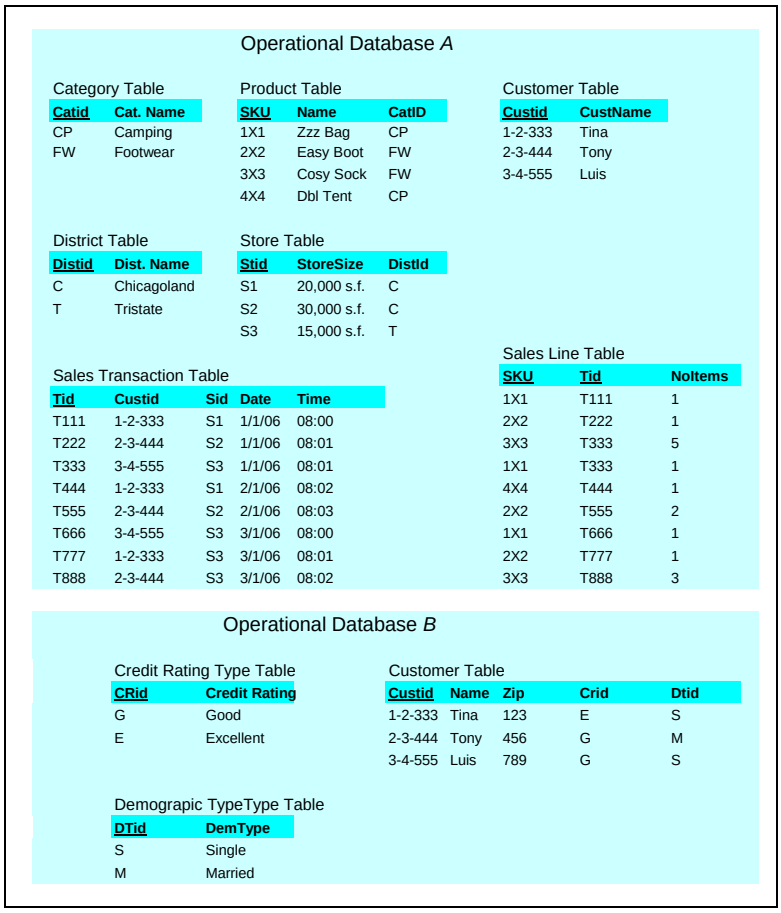


Figure 2a: ER Modeled Operational Database



Operational Database *A*

Category Table

Catid	Cat. Name
CP	Camping
FW	Footwear

Product Table

SKU	Name	CatID
1X1	Zzz Bag	CP
2X2	Easy Boot	FW
3X3	Cosy Sock	FW
4X4	Dbl Tent	CP

Customer Table

Custid	CustName
1-2-333	Tina
2-3-444	Tony
3-4-555	Luis

District Table

Distid	Dist. Name
C	Chicagoland
T	Tristate

Store Table

Stid	StoreSize	DistId
S1	20,000 s.f.	C
S2	30,000 s.f.	C
S3	15,000 s.f.	T

Sales Transaction Table

Tid	Custid	Sid	Date	Time
T111	1-2-333	S1	1/1/06	08:00
T222	2-3-444	S2	1/1/06	08:01
T333	3-4-555	S3	1/1/06	08:01
T444	1-2-333	S1	2/1/06	08:02
T555	2-3-444	S2	2/1/06	08:03
T666	3-4-555	S3	3/1/06	08:00
T777	1-2-333	S3	3/1/06	08:01
T888	2-3-444	S3	3/1/06	08:02

Sales Line Table

SKU	Tid	NoItems
1X1	T111	1
2X2	T222	1
3X3	T333	5
1X1	T333	1
4X4	T444	1
2X2	T555	2
1X1	T666	1
2X2	T777	1
3X3	T888	3

Operational Database *B*

Credit Rating Type Table

CRid	Credit Rating
G	Good
E	Excellent

Customer Table

Custid	Name	Zip	Crid	Dtid
1-2-333	Tina	123	E	S
2-3-444	Tony	456	G	M
3-4-555	Luis	789	G	S

Demograpic TypeType Table

DTid	DemType
S	Single
M	Married

Figure 2b: Sample Data for Operational Databases *A* and *B*

In order to enable the analysis of sales related data, a dimensionally modeled data mart *C* is created. The data model that is produced by the dimensional modeling method is known as a star-schema (Chaudhuri, 1997). A star schema for the data mart C is shown in Figure 2c. This data mart contains information from the operational databases *A* and *B*. The purpose of data mart *C* is to enable the analysis of sales quantity across all dimensions that are relevant for the decision-making process, and are based on existing and available operational data. It contains one fact table and four dimension tables. The fact table *SALES* contains a numeric measure (*Units Sold*) and foreign keys pointing to the relevant dimensions. The dimension table *Customer* integrates the data from the *Customer Table* in database *A* and all three tables in

database *B*. The dimension table *Store* merges data from tables *Store* and *District* in database *A*. The dimension table *Product* merges data from tables *Product* and *Category*, also from the database *A*. The dimension table *Calendar* contains details for each date that fits the range between the date of first and last transaction recorded in the *Sales Transaction Table* in database *A*.

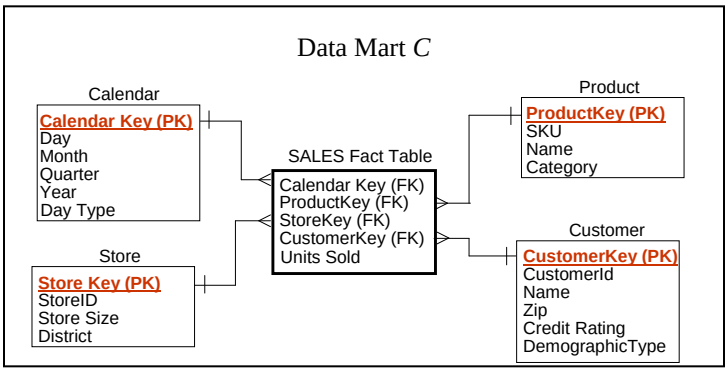


Figure 2c: Dimensionally Modeled Data Mart *C*

Each dimension has a new key, specially designed for the dimension itself. As shown in Figure 2d, the values of the keys are not imported from the operational database. Instead, a value of a key is a unique system-generated semantic-free identifier. This feature insulates dimensions from possible changes in the way operational keys are defined (and possibly re-defined) in operational databases over time. The system-generated key also has a role in tracking the history of changes in a dimension's records.

Note that the existence of the system-generated identifier does not eliminate the values of the operational keys in the data-mart tables (see, in Figures 2c and 2d, SKU in Product dimension, CustomerID in Customer dimension, and StoreID in Store dimension). These values allow the data to be related back to the operational systems. For the same reason, the Tid (transaction identifier) column can be included in the sales fact table. For simplicity and readability reasons, we are not showing the Tid column in Figures 2c and 2d.

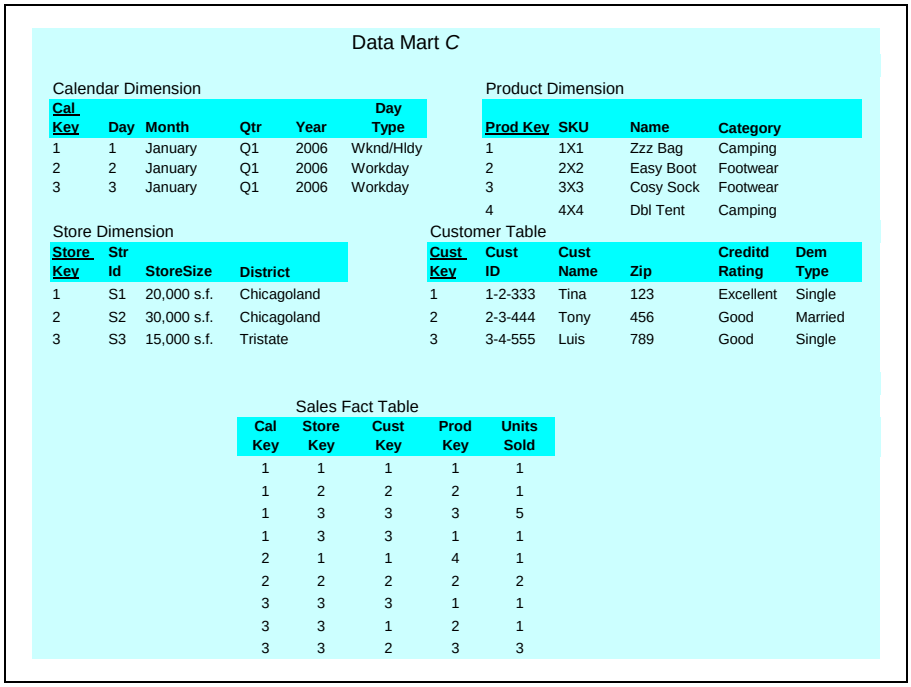


Figure 2d: Sample Data for Data Mart *C*

Once the data mart *C* is modeled using dimensional modeling techniques, and populated with the data from databases *A* and *B*, finding answers to questions such as *"Find the top ten products sold in stores of 20,000 s.f. or higher, to the customers with Excellent credit rating during the month of January for the past four years"* can be achieved in a quick fashion by issuing one simple query. If the data mart *C* were not developed, the process of finding an answer to this question would be much more complicated and would involve rummaging through the operational databases *A* and *B*, and issuing multiple queries, which would then have to be merged.

Once a dimensionally modeled data mart is in place, performing data analysis is fairly straightforward. It involves using OLAP tools, which allow users to query fact and dimension tables by using simple point-and-click query-building applications. There are several architectural approaches for developing a data warehouse, but they all give the end-user a dimensionally modeled data mart as the conceptual interface (Jukic, 2006). Figure 3 illustrates two common architectural options.

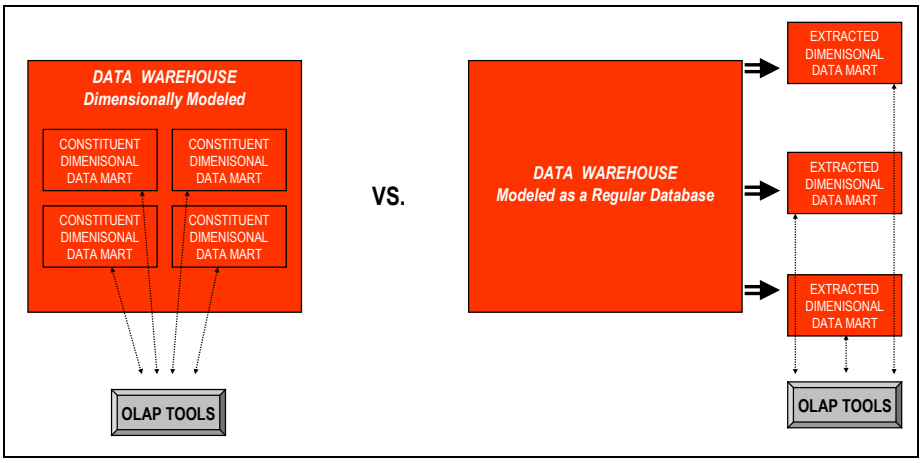


Figure 3: OLAP Tool as an Interface to Two Different Data Warehouse Architectures

One option, shown on the left side of Figure 3, is a data warehouse as a composition of various intra-connected dimensionally modeled data marts (Kimball, 1998). In that case, OLAP tools can be connected directly to the data warehouse. Another option, shown on the right side of Figure 3, models a data warehouse as a regular relational database with the dimensional data marts extracting data from the data warehouse (Inmon, 2002). The data marts present data to the users in the OLAP-friendly dimensional way.

Regardless of which architectural option is used, OLAP tools are used as a way to interface with the data warehouse. The next section describes the particular analytical features that OLAP tools offer to the users.

3. OLAP Functionalities

There are numerous OLAP tools available in the market today. This section gives an overview of functionalities that are common for all OLAP tools. The three basic OLAP features that are used regularly by analysts are commonly referred to as:

- ***Slice and Dice***
- ***Pivot (Rotate)***
- ***Drill Down and Drill Up***

A running example is used to illustrate these operations. The example is based on the Data Mart *C* illustrated by Figures 2c and 2d. Figure 4a shows the result of the following query on the Data Mart C: "*For each individual store, show separately the number of product units sold for each product category during workdays and during holiday/weekend days*." Figure 4a displays how a typical OLAP tool would display the result of this query. The results shown are actual results based on the data from Figure 2d.

	Sales - Units Sold	
	Camping	**Footwear**
Store 1		
Work Day	1	0
Weekend/Holiday	1	0
Store 2		
Work Day	0	2
Weekend/Holiday	0	1
Store 3		
Work Day	1	4
Weekend/Holiday	1	5

Figure 4a: Query Result Set in OLAP Tool

Specifying this query in an OLAP tool is quite simple. An interface resembling Figure 2c would be given to the user and the user would simply choose (e.g. via the drag and drop method) which attributes from which dimensions to use in the query. For the query listed above, the user would drag (from the graphical interface representing the schema in Figure 2c) and drop (on the graphical interface representing the query constructions space) the attribute *Store ID* from dimension *Store* and attribute *Day Type* from dimension *Calendar* on the vertical axis. On the horizontal axis the user would drop the attribute *Category* from the dimension *Product*, and in the result area, the attribute *Units Sold* from the *Sales* fact table. Once a query has displayed the results on a computer screen (Figure 4a), the user now has the option to perform any of the three above-listed basic OLAP operations.

Figure 4b shows the result of a **Slice and Dice** operation performed on the query shown in Figure 4a. The Slice and Dice operation simply adds, replaces or eliminates specified dimension attributes (or part of the dimension attributes) from the already displayed result. For the result shown in Figure 4b, the user specified that only the results for *Camping* products sold on *Workday*s should be displayed. In other words,

in Figure 4b the results showing sales of *Footwear* products and sales on *Weekend/Holiday* days were "sliced-out" from the original query (Figure 4a). The new query is now: "*For each individual store show the number of product units sold for the Camping product category during workdays."*

	Sales - Units Sold	
	Camping	
Store 1		
Work Day	1	
Store 2		
Work Day	0	
Store 3		
Work Day	1	

Figure 4b: Slice and Dice First Example

Even though the name of this operation is Slice and Dice, the operation can actually replace or add dimension attributes. In the next example, shown in Figure 4c, the user modified the query shown in Figure 4a by replacing the Category attribute (from the Product dimension) with the Credit Rating attribute (from the Customer dimension). The wording of this modified query is: "*For each individual store show separately the number of product units sold to customers with different credit rating values during workdays and during holiday/weekend days.*"

	Sales - Units Sold	
	Good (Credit Rating)	**Excelent (Credit Rating)**
Store 1		
Work Day	0	1
Weekend/Holiday	0	1
Store 2		
Work Day	2	0
Weekend/Holiday	1	0
Store 3		
Work Day	4	1
Weekend/Holiday	6	0

Figure 4c: Slice and Dice Second Example

The next example, shown in Figure 4d, illustrates the **Pivot** (or **Rotate**) operation. Unlike the Slice and Dice operation, the Pivot operation does not change the values displayed in the original query, it simply

reorganizes them. In the case of the query shown in Figure 4d, the Product Category attribute and the Store ID attribute simply swapped their axes. Because the pivot action does not change the values shown to the user, the wording for the queries shown in Figures 4a and 4d is the same: "*For each individual store show separately the number of product units sold for each product category during workdays and during holiday/weekend days.*" In Figure 4a the Product Category was placed on the horizontal axis and Store ID was placed on the vertical axis. In Figure 4d, the pivoting action was performed and Product Category was rotated onto the vertical axis, whereas Store ID was moved to the horizontal axis.

	Sales - Units Sold		
	Store 1	**Store 2**	**Store 3**
Camping			
Work Day	1	0	1
Weekend/Holiday	1	0	1
Footwear			
Work Day	0	2	4
Weekend/Holiday	0	1	5

Figure 4d: Pivot Example

The final example in this section illustrates **Drill Down** and **Drill Up** operations. The purpose of these operations is to increase (in the case of Drill Down) or decrease (in the case of Drill Up) the granularity of the data shown in the query result. Figure 4e illustrates the result of a Drill Down operation performed on a query shown in Figure 4a. In the example shown, the user decided to drill down in the Product dimension from product Category to product Name. The wording of the query whose result is shown in Figure 4a is expanded in the following way: "*For each individual store, show separately the number of product units sold for each product category, and within each product category for each individual product name, during workdays and during holiday/weekend days.*" The Drill Down operation allows users to drill through hierarchies within dimensions. A hierarchy is a set of nested levels. Another way to say this is that an item at any level of a hierarchy is related to (possibly) many items at a lower level, but no more than one item at any higher level. Consider these items in the *Customer* table: *customer ID*, *customer name*, and *zip*. This would be a hierarchy because one ID has one name and one zip. However, one zip has many customer names and one customer name can have many IDs associated with it (e.g. we can have several customers with the same name, say, John Smith). A drill hierarchy allows the user to expand a value at one level to

show all the detail below it, or to collapse detail to show only the higher level value. Thus, John Smith can be expanded to show all IDs under that name, or IDs could be merged into equivalent names or zipcodes. Some dimensions can have more than one drill hierarchy. Consider the dimensions in the Data Mart *C*, shown in Figure 2c. For example, the *Store* dimension has two hierarchies: *Store ID–Store Size* and *Store ID–Store District*, while the *Product* dimension has only one hierarchy: *SKU–-Name–Category*. Because each product name belongs to exactly one product category and each product category contains multiple product names, user can drill down from the product category to the product name. Consequently, a user can Drill Up from the product name to the product category. To illustrate the Drill Up operation we can simply consider Figure 4e as a starting query and look at the Figure 4a as the product of a Drill Up operation from product Name to product Category. In most OLAP tools, Slice-and-Dice, Pivot, and Drill Down/Up actions are implemented in a straightforward manner, usually using some form of point-and-click and drag-and-drop methods.

	Sales - Units Sold			
	Camping		**Footwear**	
	Zzz Bag	Dbl Tent	Easy Boot	Cosy Sock
Store 1				
Work Day	0	1	0	0
Weekend/Holiday	1	0	0	0
Store 2				
Work Day	0	0	2	0
Weekend/Holiday	0	0	1	0
Store 3				
Work Day	1	0	1	3
Weekend/Holiday	1	0	0	5

Figure 4e: Drill-Down Example

Closer analysis of the above examples reveals that the dimensional model is essential for OLAP. If the underlying data was not organized in a dimensional way, with a (usually numeric) fact table in the center connected to a number of (usually textual) dimension tables, the three basic OLAP operations could not be effectively and properly performed.

Other than the ability to perform Slice-and-Dice, Pivot and Drill Down/Up, many other capabilities are found in various contemporary OLAP tools. For example today's OLAP tools are able to create and examine calculated data; determine comparative or relative differences; perform exception analysis, trend analysis, forecasting, and regression analysis, as well as number of other useful analytical functions. However, those functionalities are found in other non-OLAP applications, such as statistical tools or spreadsheet software. Contemporary spreadsheet software even has some rudimentary capacity of performing basic OLAP functions, on the limited amount of data it can store. What really distinguishes OLAP tools from other applications is the capability to easily interact with the dimensionally modeled data marts and data warehouses and, consequently, the capability to perform OLAP functions on large amounts of data.

4. Relational versus Multidimensional Data Model

The previous section described the functionalities of an OLAP tool. This section describes two different database models on which the architecture of OLAP tools is based. The first model described is the standard Relational Database Model. This model is a basis for the contemporary Relational Database Management Systems (RDMBS) which are used to implement the vast majority of today's operational corporate databases. Examples of RDBMS software include Oracle, IBM DB2, MS SQL Server, and NCR Teradata (NCR Teradata was developed specifically to accommodate large data warehouses, whereas the other ones are used both for hosting operational databases as well as data warehouses). In the relational database model, the database is a collection of two-dimensional tables, where each row of the table represents a database record. Figure 2a shows diagrams for two relational operational databases and Figure 2b shows the populated tables for those two relational databases. Figure 2c shows a dimensional model that is implemented as a relational database. Figure 2d shows populated tables for the dimensional model shown in Figure 2c.

In order to describe the other model, the Multidimensional Database Model, and contrast it with the Relational Model, we will use a simplified example shown in Figure 5a.

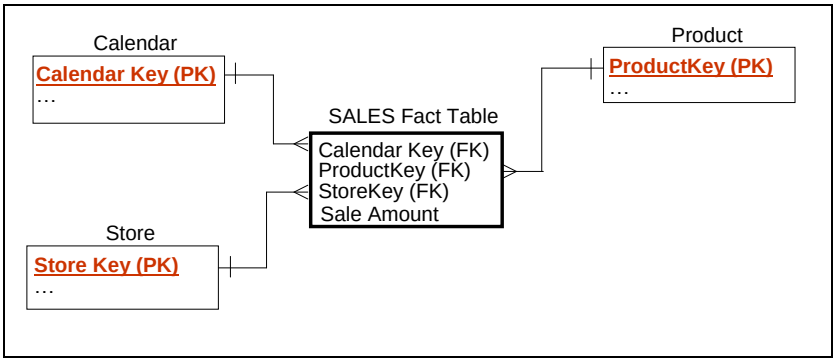


Figure 5a: Simple Data Mart

This example uses a simple data mart with three dimensions and one fact table. The star-schema in Figure 5a is equivalent to the star-schema in Figure 2c, with two small differences. It uses three dimensions (instead of four), and the fact table stores the sale *amount* (instead of number of units sold). Figure 5b shows the relational implementation of the fact table from Figure 5a. The fact table contains the keys of all dimension tables to which it is connected, and the numeric fact *Sale Amount*. Each record corresponds to one instance of a *Product* being sold in a specific *Store* on a certain *Day* for an *Amount*.

Sales Fact Table

CalendarKey	**ProductKey**	**StoreKey**	Sale Amount
1	**1**	**1**	**$100**
1	**2**	**1**	**$120**
1	**3**	**1**	**$200**
.	.	.	.

Figure 5b: Relational Implementation of a Fact Table

A multidimensional Database Model can be visualized as cube with a number of dimensions. In fact a cube can have more than three dimensions, but in order to be able to give a visual example in this paper we use an example with exactly three dimensions. The cube acts as a multidimensional array in a conventional programming language. The space for the entire cube is pre-allocated and to find or insert data, the dimension values are used to calculate the position. A multidimensional database uses a multi-cube storage design, since it can contain many cubes. Figure 5c shows the multidimensional implementation of the fact

table from Figure 5a. Each cell in the cube corresponds to the one instance of a *Product* being sold in a specific *Store* on a certain *Day* for an *Amount* (for visibility purposes, only one cell with the amount of $100 is shown). The key difference between the two models is the search method. In the relational model, in order to locate a record, some type of search has to take place on the fact table. The speed of the search depends on issues such as how the records are sorted or is the table indexed. In the multidimensional cube every record can be looked up directly, eliminating the need for a search. This is because each cell has a direct address composed of the values of the dimension's attributes, e.g. (1,1,1) → $100).

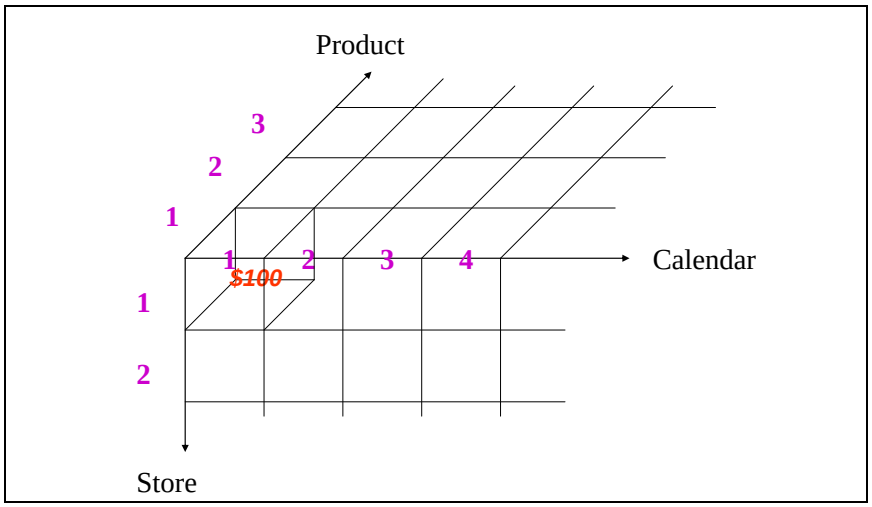


Figure 5c: Multidimensional Implementation of a Fact Table

5. OLAP Categories

There are several different categories of OLAP tools, depending on which database model is used. The **MOLAP** (multidimensional online analytical processing) engine takes the data from the warehouse or from the operational sources. The MOLAP engine then stores the data in proprietary data structures, multi-dimensional cubes. The complexity of the underlying data is hidden from the MOLAP tool user. In other words, they perform standard OLAP functions without having to understand how the cubes are formed and how they differ from relational tables. A typical MOLAP architecture is shown in Figure 6a.

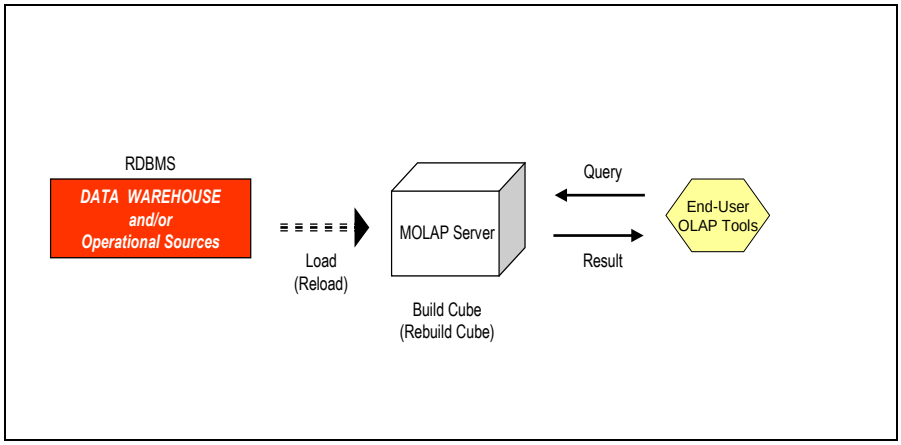


Figure 6a: Typical MOLAP Architecture

Generally, a separate MOLAP server containing a limited amount of data is used. The main characteristic of MOLAP is that it provides very fast analysis. The way the MOLAP server achieves this goal is that it pre-calculates as many outcomes as possible and stores them in cubes. It pre-calculates the hierarchies within the individual dimensions as well as the intersections between attributes of multiple dimensions.

It is important to note while MOLAP cubes perform very well when analyzing aggregated data, they are not appropriate for holding transaction level detail data. The transaction level detail data is the finest granularity data where each record corresponds to a single real-world transaction. E.g. a record stating that a *sleeping bag ZZZ Bag was sold to the customer Tina in the store S1 on January 1, 2006*, is a transaction level detail record. On the other hand, in aggregated data, one record reflects a summary of more than one transaction. E.g. a record stating that *in Q1 of 2006, 200 units of products from Camping category were sold to the customers with Excellent credit rating in Chicagoland stores* is an aggregated data record. It was created by summarizing all the transaction level records containing data about each individual sale of Camping products to Excellent credit rating customers in Chicagoland stores during the first quarter of 2006. Cubes are limited in space and in a typical corporation the sheer amount of transaction level data would surpass the capacity of a cube. Even if the transaction level data would somehow fit into the cube,

other issues, such as data sparsity (most dimension attribute intersections are empty in the case of transaction level data) make cubes inappropriate for dealing with non-aggregated data.

MOLAP is very fast in the execution of queries, due to the fact that each cell in a cube has a direct address and answers to a large portion of common queries are pre-calculated. The calculation engine can create new information from existing data through formulas and transformation. Pre-aggregated summary data and pre-calculated measures enable quick and easy analysis of complex data relationships. Often the query "processing" part boils down to a direct data lookup. While MOLAP performs very well when it comes to data retrieval, the updating of a cube can be quite slow. Data loading can take hours, and the cube calculation can take even more time (every time new data is available, the cube has to be re-loaded and re-calculated.) However, the speed with which analytical queries can be answered is often a much more important factor than the speed of loading the new data and creating an updated cube.

In typical cases, data is loaded into MOLAP servers from data warehouses hosted on relational DBMS (RDBMS) platforms. However, there are instances in practice when data is loaded into cubes directly from the operational data sources to satisfy current-data analysis needs.

Another important category of OLAP tools are relational OLAP tools, commonly referred to as **ROLAP** tools. A high-level view of a typical ROLAP architecture is shown in Figure 6b. The ROLAP tool provides the same common OLAP functionalities. Queries are created in a standard point-and-click way. The ROLAP server translates the queries into SQL (Structured Query Language), the standard query language for all contemporary RDBMS systems. The SQL version of the query is sent to the data warehouse hosted on the RDBMS platform. The query is executed in the RDBMS and the resulting data set is sent to the ROLAP server and then to the End-User OLAP tool which presents them to the user in a form similar to what is shown in Figures 4a-4e.

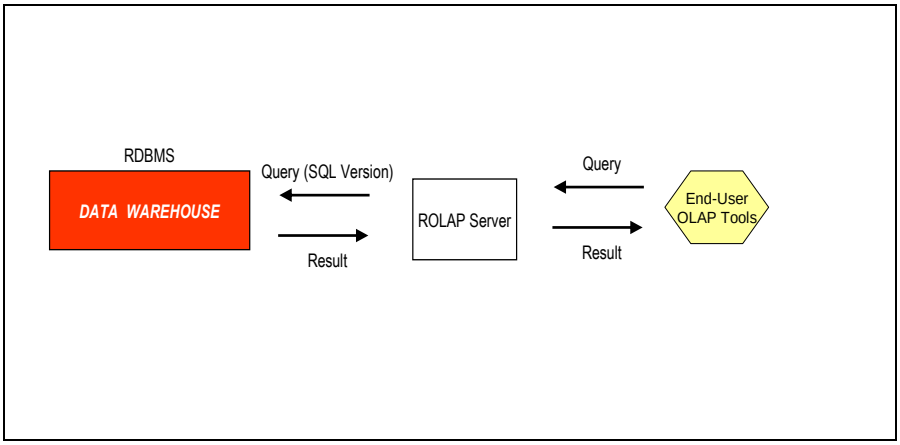


Figure 6b: Typical ROLAP Architecture

The ROLAP architecture imposes no limitations on the size of the database or the kind of analysis that may be performed. However, due to the fact that results are not pre-calculated the performance of queries is not as fast as with MOLAP tools.

The tradeoff of MOLAP vs. ROLAP is performance vs. storage. Queries are executed faster with MOLAP tools, but ROLAP is capable of handling much larger quantities of data which makes it suitable for processing transaction-level detail data. Also, the continuous advances in the speed of query processing with RDBMS software are shrinking the performance gap between MOLAP and ROLAP tools.

The hybrid online analytical processing (HOLAP) architecture combines MOLAP and ROLAP approaches. The typical HOLAP architecture is shown in Figure 6c. HOLAP aspires to take advantage of the strengths of both methods. In a hybrid solution, for example, the relational database can be used to store the bulk of the detail data and the multidimensional model can be used to store summary data.

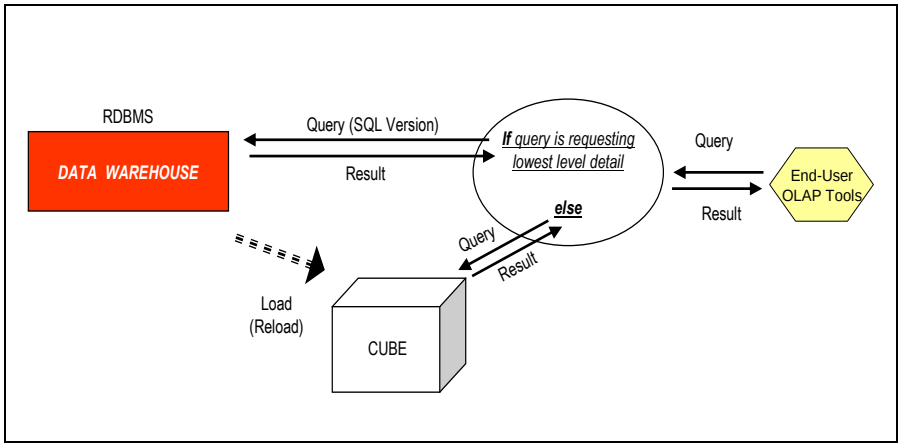


Figure 6c: Typical HOLAP Architecture

Even though HOLAP technology is able to provide solid performance, even when analyzing large amounts of data, HOLAP implementations typically do not achieve all the strengths indigenous to MOLAP and ROLAP approaches. In a sense, HOLAP technology is as much a compromise as it is a synergy of the two approaches (Gonzales, 2006).

DOLAP is another term that is often used when OLAP architectures are discussed. DOLAP stands for Desktop OLAP. Like MOLAP tools, DOLAP tools also use multidimensional cubes. The difference is that the cubes used by DOLAP are actually downloaded to the individual computers of the end users where all of the query processing actually takes place. Such cubes are much smaller than the ones used by MOLAP tools. These relatively small cubes (also referred to as micro-cubes) can be easily replicated and distributed to multiple users. They are easy and convenient to use (e.g. a user can perform analysis on a laptop while disconnected from the network), but they have limited functionality. This is not only due to the small amount of data these cubes can accommodate, but also due to the fact that the information in these cubes is static (there are no updates from the data warehouse once the cube is created.)

6. Summary

Within all areas of business, the role of analytical data repositories, such as data warehouses and data marts, continues to grow. As a result, additional large sets of clean, integrated data are being created. It is

only natural that a business with large repositories of clean and integrated analytical data would take advantage of this data to make better decisions. This chapter has shown how OLAP is tied to such data repositories. The basic OLAP functions give the user an efficient way to access that data to support managerial decision making. As OLAP has grown in importance, various tool deployment options have developed, such as web-based, client-server, or desktop-standalone.

OLAP tools are now an essential part of the decision making process for every organization that collects large amounts of data. The more data accumulated in its operations, the more essential OLAP capabilities become to an organization. OLAP applications are found today in widely divergent business areas; such as finance, sales, marketing, or manufacturing (Begg, 2007). OLAP applications today offer a variety of available features and interfaces that can serve the analytical needs of users with different demands and levels of sophistication, such as parametric users, casual ad-hoc users, or business analysts. The universal appeal of OLAP is in the simplicity of structure and conceptualization, and consequently, in its straightforward usability. Naturally, OLAP has its limitations. More extensive analysis of the data can be accomplished with more complex methods, such as statistical procedures or data mining. However, in most business-related scenarios, such complex methods should be applied only after the OLAP based analysis is undertaken. In fact, the results of OLAP analysis, in addition to revealing immediately applicable findings that have significant impact, often provides the direction and basis for applying additional procedures (if they are needed).

One of the more descriptive definitions of the OLAP process was given by Gonzalez in (2006) where he defines it as a process of data interrogation: "*Online analytical processing is all about the interrogation of a data domain. The approach to data interrogation begins with the broadest questions being asked across the highest aggregated data. For example, give me this month's sales across the entire US retail chain and for all product groups. As the analyst begins to dive deeper into the data, their questions become increasingly specific. In other words, a follow-up question to the one above might be, give me this month's sales for only the Northeast Region for the US chain for all product groups*." In other words, OLAP tools

allow users to investigate the data both in its aggregated and in its detailed form. Often the answers to the queries on data in the aggregated form lead to new questions on the data that is in more detailed form. The OLAP functions enable the user to move from one level of aggregation to another easily, in an intuitive fashion. This not only gives the user access to the data facts, but the ability to view the data easily and in multiple ways helps to understand business realities that the data captures. Knowing the facts, and properly analyzing and interpreting them, leads to a fact-based decision making process. A fact-based decision making process is obviously superior to any other conceivable alternative, and OLAP represent one of its most important means of supporting this process available today.

References

Begg, C., T. Connolly, and R. Holowczak, *Business Database Systems*. Reading, MA: Addison Wesley, 2007.

Chaudhuri, A. and U. Dayal, "An Overview of Data Warehousing and OLAP Technology," *ACM SIGMOD Record*, 26, 1, 1997, 65-74.

Chenoweth, T., D. Schuff, and R. St. Louis, "Method for Developing Dimensional Data Marts," *Communications of the ACM*, 46, 12, 2003, 93-98.

Gonzales, M. L., *Hands-On OLAP*, Seattle, WA: The Data Warehousing Institute, 2006.

Inmon, W., *Building the Data Warehouse, 3rd Edition*, New York, NY: Wiley, 2002.

Jukic, N., "Data Modeling Strategies and Alternatives for Data Warehousing Projects," *Communications of the ACM*, 49, 4, 2006, pp. 83-88.

Kimball, R., L. Reeves, M. Ross, and W. Thornthwaite, *The Data Warehouse Lifecycle Toolkit*, New York, NY: Wiley, 1998.

Index Terms

OLAP
ROLAP
DOLAP
MOLAP
Slice-and-Dice
Pivot
Hierarchical
Drill Up
Drill Down
Data Warehouse
Data Mart
Dimensional Model

Profiles

Nenad Jukic (njukic@luc.edu)
Dr. Nenad Jukic is an Associate Professor of Information Systems and the Director of the Graduate Certificate Program in Data Warehousing and Business Intelligence at Loyola University Chicago. Dr. Jukic conducts active research in various information technology related areas, including data warehousing/business intelligence, database management, e-business, IT strategy, and data mining. His work has been published in a number of information systems and computer science journal and conference publications. Aside from academic work his engagements include projects with U.S. military and government agencies as well as consulting for corporations that vary from startups to Fortune 500 companies.

Boris Jukic (bjukic@clarkson.edu)
Dr. Boris Jukic is an Associate Professor of Management Information Systems at Clarkson University. His research interests include IT strategy, decision support systems, and the application of economic theory in various areas of Information Technology, ranging from computer network resource management to the application of new database technologies in Electronic Commerce. His research appeared in various engineering, management and economics publications and conference proceedings

Mary Malliaris (mmallia@luc.edu)
Dr. Mary E. Malliaris is an Associate Professor in Information Systems at Loyola University Chicago. Her research and teaching interests are in databases, data warehousing, data mining and database marketing. She has published articles using neural networks in The Global Structure of Financial Markets, The International Journal of Computational Intelligence and Organizations, Neural Networks in Finance and Investing, Neurocomputing, Neurovest, and Applied Intelligence among others. She has served as a reviewer for The Financial Review, The Journal of Applied Business Research, Mitchell-McGraw Hill Publishers, Wall Data, and ISECON. She is currently the production editor for The Journal of Economic Asymmetries.