Concepts and Fundaments of Data Warehousing and OLAP

Book · January 2017

CITATION
1

READS
23,411

1 author:

Fernando Almeida
Instituto Superior Politécnico Gaya
117 PUBLICATIONS **359** CITATIONS

SEE PROFILE

Concepts and Fundaments of Data Warehousing and OLAP

Abstract

In recent years, it has been imperative for organizations to make fast and accurate decisions in order to make them much more competitive and profitable. Data warehouses appear as key technological elements for the exploration and analysis of data, and subsequent decision making in a business environment. This book deals with the fundamental concepts of data warehouses and explores the concepts associated with data warehousing and analytical information analysis using OLAP. The reader is guided by the theoretical description of each of the concepts and by the presentation of numerous practical examples that allow assimilating the acquisition of skills in the field.

10/8/2017

Fernando Almeida, PhD. INESC TEC and University of Porto

Table of Contents

Acronyms

BI - Business Intelligence

CPU - Central Processing Unit

DM - Data Mining

DOLAP - Desktop OLAP

DSS - Decision Support System

DW - Data Warehouse

ETL - Extract, Transform, Load

HOLAP - Hybrid OLAP

I/O - Input/Output

IS - Information Systems

IT - Information Technology

JOLAP - Java OLAP

MOLAP - Multidimensional OLAP

NF - Normal Form

OLAP - OnLine Analytical Processing

OLTP - OnLine Transaction Processing

ROLAP - Relational OLAP

SOLAP - Spatial OLAP

SQL - Structured Query Language

Glossary

Dimension - a structure that categorizes facts and measures in order to enable users to answer business questions. Commonly used dimensions are people, products, place and time. In a data warehouse, dimensions provide structured labeling information to otherwise unordered numeric measures.

DM - the process of sorting through large data sets to identify patterns and establish relationships to solve problems through data analysis.

ETL - refers to a process in database usage and especially in data warehousing.

Fact table - consists of the measurements, metrics or facts of a business process. It is located at the center of a star schema or a snowflake schema surrounded by dimension tables.

Metadata - summarizes basic information about data, which can make finding and working with particular instances of data easier.

Normalization - the process of organizing the columns (attributes) and tables (relations) of a relational database to reduce data redundancy and improve data integrity.

Transaction - sequence of operations performed as a single logical unit of work. A logical unit of work must exhibit four properties, called the atomicity, consistency, isolation, and durability (ACID) properties, to qualify as a transaction.

Trigger - special kind of stored procedure that automatically executes when an event occurs in the database server. DML triggers execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

1. Introduction

1.1 Contextualization

Nowadays, automated information technologies are usually repetitive or administrative. We understand information systems for operational applications, those that meet the operational needs of the company. In these systems, the most important concepts are the up-to-date information and the response time. Once the most pressing operational needs are met, there is a new set of needs of the company's systems, which they qualify as informational needs. For the information needs, we mean those that seek to obtain the necessary information as the basis for decision making, both strategic and tactical. These information requirements are largely based on the analysis of a huge number of data, which is just as important to obtain a very detailed commercial value as the total value for it. There is also a crucial historical view of all variables analyzed, and the analysis of environmental data. These requirements are not, a priori, difficult to solve because information is actually in the operating systems. Any activity undertaken by the company is completely reflected in their databases.

The reality, however, is different, because meeting the needs of manufacturers of such information systems is faced with various problems. Firstly, massive information queries (in order to obtain the value relation, or set of grouped request values), can be negatively affected by the level of service from other systems, since the queries we are talking about tend to be very expensive in features. In addition, business needs are met with limited flexibility to look for information and its inconsistency, due to the lack of global vision (particular view of each one the data are stored in the operating system that executes it).

In this situation, the next evolutionary step has to be the generation of a dual operating environment, which is commonly called the Information Center, in which information is updated much less frequently than in operational environments and requirements at the level of service users are More flexible. This strategy solves the problem of planning features and applications that require a high level of service using the operating environment and requiring huge data queries that work in the Information Center.

But the problems do not end here. The information remains the same structure in operational applications, such as query must access a multitude of places to obtain the desired dataset. The response time to requests for information is very high. In addition, when performing different information systems with different views and different goals, it is often not possible to obtain the desired information in an easy and does not have the necessary reliability.

In fact, most database management systems are transactional, which means, they are optimized for data insertion and updating, and are not intended to function as decision support systems. When an analyst wants to obtain information from a transactional system, it is necessary to summarize or derive the data. These data transformation operations are necessary because most of the information sought is not directly available in the transactional systems, although the cost of these operations can be obtained. The same is no longer the case in data warehouses, since they are designed specifically for decision support, being

possible to find many fields of information that the transactional systems do not have. In fact, a data warehouse can integrate multiple database transaction systems. Data mining is often considered the next step after the implementation of a data warehouse, due to the integration and transformation of data that it requires, which is not the case in traditional transaction systems. There is no doubt that the existence of a data warehouse facilitates the conduction of data mining studies, so it appears as a natural sequence of the previous one.

1.2 Objectives

This mini book intends to provide a brief reference guide for undergraduate students that want to learn data warehousing and OLAP. The book presents the main concepts and elements of data warehousing. Additionally, the process of analyzing information and performing analytical functions using OLAP is detailed.

1.3 Book Structure

The book is organized in just two concise chapters. The first chapter "Data Warehousing" presents the evolution of the OLTP concept in a Data Warehousing (DW) environment. It presents the ETL process for the migration of data and the most common DW architectures. Finally, the second chapter "OLAP" presents the characteristics of this language and its main operators. These OLAP functions are presented using practical examples.

2. Data Warehousing

2.1 Overview

The data warehouse (DW) acts as a central repository of information originating from one or more data sources. Data flows from transactional systems and other relational databases to the data warehouse and generally consist of structured, semi-structured, and unstructured data. This data is loaded, processed and consumed on a regular basis. Users such as data scientists, business analysts, and decision makers use business intelligence tools, SQL clients, and spreadsheets to access data processed in the data warehouse.

Figure 1 provides a global vision of a DW environment. Some important elements of a DW environment can be already seen, such as the ETL process, metadata, data marts and OLAP tools. These elements will be detailed in the next sections.

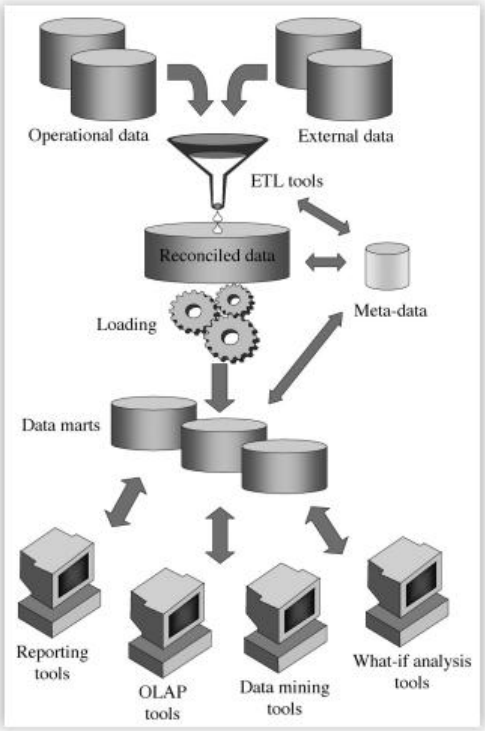


Figure 1 - Global vision of a DW environment (Rizzi, 2009)

A database which is built for on line transaction processing, OLTP, is generally regarded as unsuitable for data warehousing as they have been designed with a different set of needs in mind (i.e., maximizing transaction capacity and typically having hundreds of tables in order not to lock out users etc.). Data warehouses are interested in query processing as opposed to transaction processing. Figure 2 provides a comparative analysis between OLPT and data warehousing environments.

	OLTP	Data Warehouse
Purpose	Run day-to-day operations	Information retrieval and analysis
Structure	RDBMS	RDBMS
Data Model	Normalised	Multi-dimensional
Access	SQL	SQL plus data analysis extensions
Type of Data	Data that runs the business	Data that analyses the business
Condition of Data	Changing, incomplete	Historical, descriptive

Figure 2 - Comparative analysis between OLTP and data warehousing (Rea)

2.2 Definition

A data warehouse integrates and manages the flow of information from enterprise databases. By definition, it possesses the following properties:

- Oriented to subject: organized according to different business visions;
- Integrated: from heterogeneous data sources;
- No volatile: always inserted, never deleted;
- Variant in time: historical positions of activities in time.

There are several slightly different definitions of data warehousing, namely:

- A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process (1keydata);
- Data warehousing is a technology that aggregates structured data from one or more sources so that it can be compared and analyzed for greater business intelligence (Informatica);
- Massive database (typically housed in a cluster of servers, or a mini or mainframe computer) serving as a centralized repository of all data generated by all departments and units of a large organization. Advanced data mining software is required to extract meaningful information from a data warehouse (Business Dictionary).

2.3 Advantages and Challenges

A data warehouse can be used to perform a wide range of tasks, namely (Mullins):

- Track, manage and improve corporate performance;
- Monitor and modify a marketing campaign;
- Review and optimized logistics and operations;
- Increase the efficiency and effectiveness of product management and development;
- Query, join and access disparate information culled from multiple sources;
- Manage and enhance customer relationships;

- Forecast future growth, needs and deliverables;
- Cleanse and improve the quality of an organization's data.

There are many benefits to deploying and effectively using a data warehouse. From an IT perspective, separating the analytical processes in a data warehouse from the operational processes in the production applications and transactions can enhance the performance of both areas. From a business perspective, a data warehouse platform can deliver a practical way to view the past without affecting the daily operations of the business. By querying and analyzing data in the data warehouse, organizations can improve operations and enable more efficient business processes, thereby increasing revenue and raising profits.

There is also a wide range of advantages of adopting data warehouses, namely (Power, 2008):

- Integrating data from multiple sources;
- Performing new types of analytical analysis;
- Reducing costs to access historical data;
- Standardizing data across the organization, having a single vision of the data;
- Improving turnaround time for analysis and reporting;
- Sharing data and allowing others to easily access data;
- Supporting ad-hoc reporting and inquiry;
- Reducing the development burden on IS/IT;
- Removing informational processing load from transaction-oriented databases.

On the other side, there are also some disadvantages and challenges resulting from the adoption of data warehouses, namely (Thakur, 2016; Business Impact, 2015):

- Time consuming preparation and implementation;
- Difficulty in integration compatibility considering the using of different technologies;
- High maintenance costs;
- Limited use due to confidential information;
- Data ownership and data security;
- Underestimation of ETL processing time;
- Inability to capture the required data;
- Increased demands of the users.

2.4 Data Marts

A data mart (DM) can be seen as a small data warehouse, covering a certain subject area and offering more detailed information about the market (or department) in question. In Figure 3 two data marts "Sales and "Product" are used to store specific information about those items.

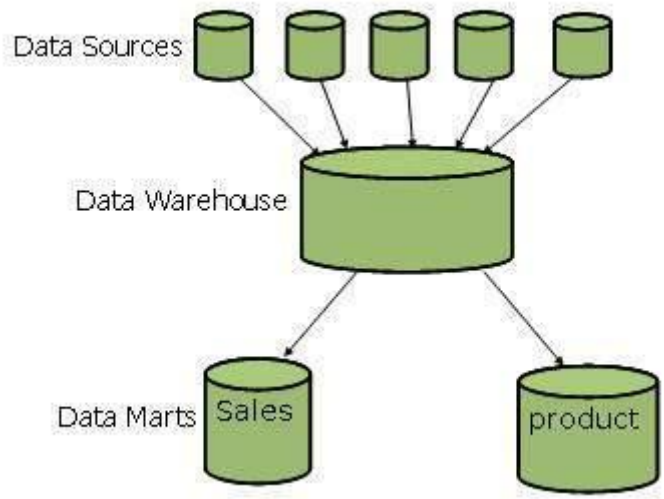


Figure 3 - Vision of data marts (Tutorials Point)

A data mart can be created in two ways:

1) Capturing data directly from transactional systems, each data mart is seeking the relevant information for its market;
2) Capturing data from all transactional systems in a central data warehouse, which in turn feed all data marts.

The first option will provide a faster data mart, but without taking into account the cross-information between the other subject areas. The second option tends to be more efficient, but it will require more time to deliver results.

A top-down perspective considers that a full, centralized DW should be developed before parts of it, summarized, can be derived in the form of Data Marts. On the other side, a bottom-up perspective considers that a DW can be composed from previously developed data marts.

There is also the possibility to build dependent or independent data marts as shown in Figure 4. In dependent data marts the data are loaded from a DW. On the other side, in the independent data marts scenario, the data to be loaded in data marts come directly from operational systems. After that, the data can be transformed to feed the DW.

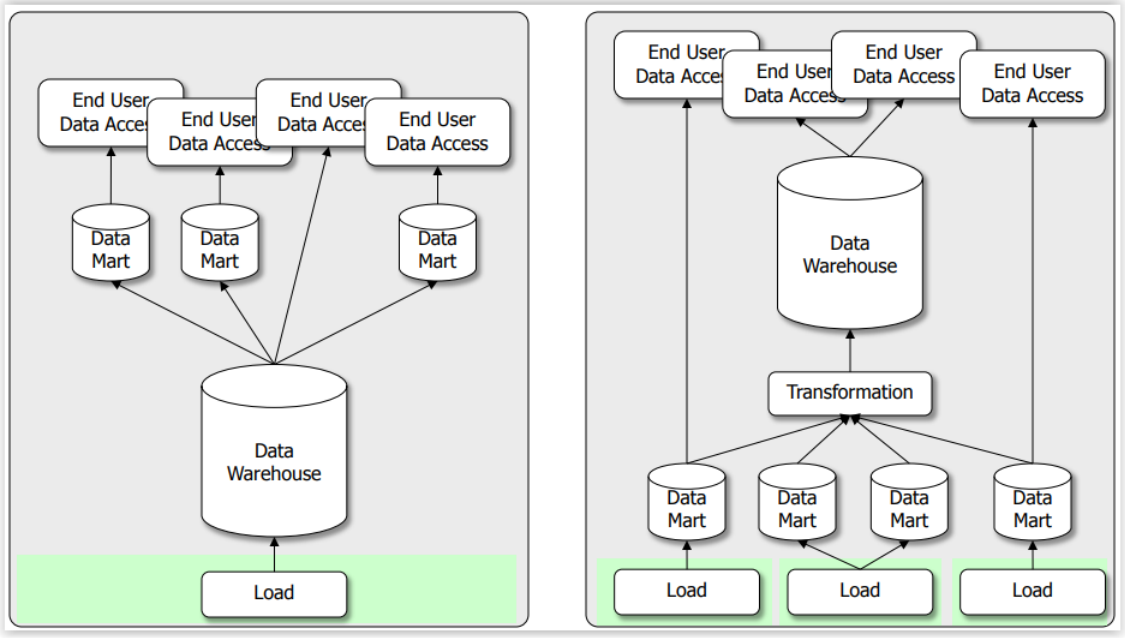


Figure 4 - Dependent vs. independent data marts (Mitschang)

There are significant differences between the DW and DM approaches. Figure 5 provides a comparative analysis.

DATA WAREHOUSE	DATA MART
◆ Corporate/Enterprise-wide	◆ Departmental
◆ Union of all data marts	◆ A single business process
◆ Data received from staging area	◆ Star-join (facts & dimensions)
◆ Queries on presentation resource	◆ Technology optimal for data access and analysis
◆ Structure for corporate view of data	◆ Structure to suit the departmental view of data
◆ Organized on E-R model	

Figure 5 - Comparative analysis between DW and DM approaches (Kumar, 2012)

Four kinds of benefits can be reached by adopting a DM approach (Algolytics):

- Streamlining of marketing and sales analyzes, report generation, prognostic model creation and management;
- Time savings considering that the analyst does not need to acquire and process data from a significant number of sources whenever a new model has to be built;
- Smaller risk of errors in the analysis, which means the results are more credible;

- Availability of the most up to date analytical information, thanks to cyclic updating of the data mart.

It is important to keep in mind that data marts can be used as a proof of concept, which is currently very important in a big project with huge budgets. In fact, implementing a data mart is much cheaper than implementing a complete DW. Additionally, data marts can co-exist with a DW. However, in order to ensure consistency and synchronization, multiple data marts should be integrated.

2.5 Metadata

In a data warehouse project, documentation is so important as the implementation process. This is because a DW project is often huge and encompasses several different areas of the organization, which means that not everyone involved is totally aware of everything that happens in the project.

Metadata is a fundamental element that belongs to the project documentation. It is used to explain other data and describe the DW environment. Metadata has the following goals:

- Resource description;
- Information retrieval and dissemination;
- Preservation and retention;
- Managing users and agents;
- Ownership and rights management.

In the context of a DW project the metadata should keep the following information:

- The structure of the data according to the view of the programmer;
- The structure of the data according to view of DSS analysts;
- The data sources of the DW;
- The transformation undergone by the data at the moment of its migration to DW;
- The data model;
- The relationship between the data model and the DW;
- The history of data extraction.

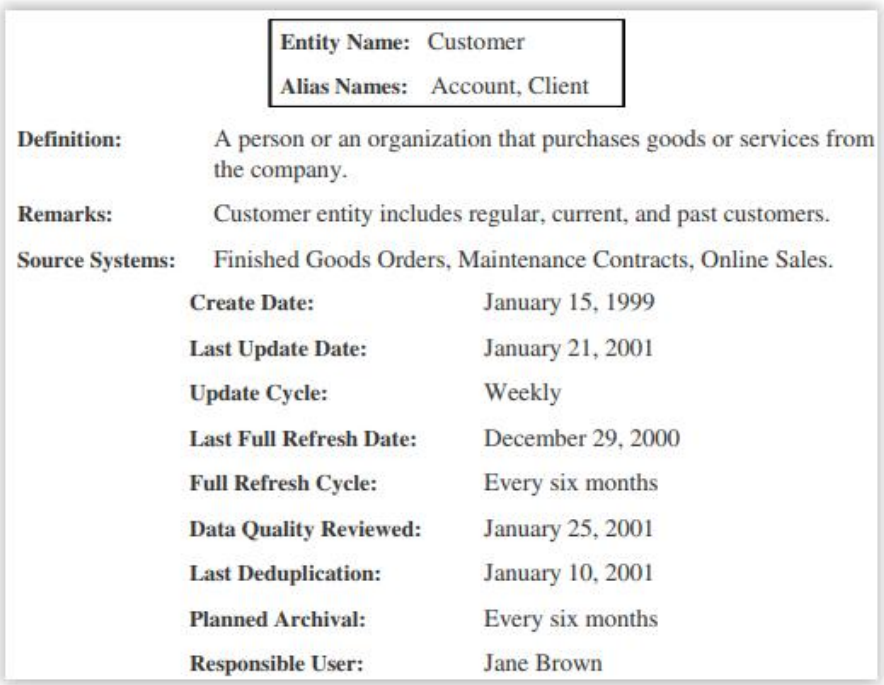
Metadata is a very important element in a DW environment. Metadata helps in driving the accuracy of reports, validates data transformation, and ensures the accuracy of calculations. Metadata also enforces the definition of business terms to business end-users. With all these uses of metadata, it also has its challenges. Some of the challenges are presented below (Tutorials Point):

- Metadata in a big organization is scattered across the organization. This metadata is spread in spreadsheets, database, and applications;

- Metadata could be present in text files or multimedia files. To use this data for information management solutions, it has to be correctly defined;

- There are no industry-wide accepted standards. Data management solution vendors have a narrow focus;
- There are no easy and accepted methods of passing metadata.

Figure 6 provides an example of a metadata file for a customer entity. This information is vital to have an idea about the origin of the data, its utility and how it is being processed.



Entity Name: Customer
Alias Names: Account, Client

Definition:	A person or an organization that purchases goods or services from the company.
Remarks:	Customer entity includes regular, current, and past customers.
Source Systems:	Finished Goods Orders, Maintenance Contracts, Online Sales.
Create Date:	January 15, 1999
Last Update Date:	January 21, 2001
Update Cycle:	Weekly
Last Full Refresh Date:	December 29, 2000
Full Refresh Cycle:	Every six months
Data Quality Reviewed:	January 25, 2001
Last Deduplication:	January 10, 2001
Planned Archival:	Every six months
Responsible User:	Jane Brown

Figure 6 - Metadata for a customer entity (Ponniah, 2001)

2.6 Approaches for Building DWs

There are generally three approaches for building a data warehouse:

1) Centralized architecture - an integrated DW, which maximizes the available processing power;
2) Federation architecture - distributing information by organizational areas;
3) Layered architecture - highly summarized data on one server, intermediate-level summarized data on a second layer, and more detailed data on a third server. The data on the 1st layer can be optimized for heavy user load and low volumes, while the other layers are best suited to handle large data volumes.

Figure 7 provides a comparative analysis of these three approaches.

Centralized architecture	Federal architecture	Tiered architecture
Only one data model	Logically consolidated	Physical central data warehouse
Performance bottleneck	Separate physical databases that store deatiled data	Separate physical databases that store summarized data
Complex to build	Faster response time	Faster response time
Easy to maintain		

Figure 7 - Comparative analysis of DW approaches

2.7 Data Architectures

There are three well known data architectures:

1) Star schema;
2) Snowflake schema;
3) Constellation schema.

Before detailing each of the architectures, there are two concepts that need to be clarified:

- Fact table - a fact table typically has two types of columns: foreign keys to dimension tables and measures those that contain numeric facts. A fact table can contain fact's data in detail or aggregated level;
- Dimension table - a dimension is a structure, usually composed of one or more hierarchies that categorizes data. If a dimension hasn't got hierarchies and levels it is called a flat dimension or list. The primary keys of each of the dimension tables are part of the composite primary key of the fact table. Dimensional attributes help to describe the dimensional value. They are normally descriptive, textual values. Dimension tables are generally smaller in size than fact table.

2.7.1 Star schema

The star schema architecture is the simplest data warehouse schema. It is called a star schema because the diagram resembles a star, with points radiating from a center. The center of the star consists of the fact table and the points of the star are the dimension tables. Usually the fact tables in a star schema are in third normal form (3NF) whereas dimensional tables are de-normalized (Datawarehouse4u.Info).

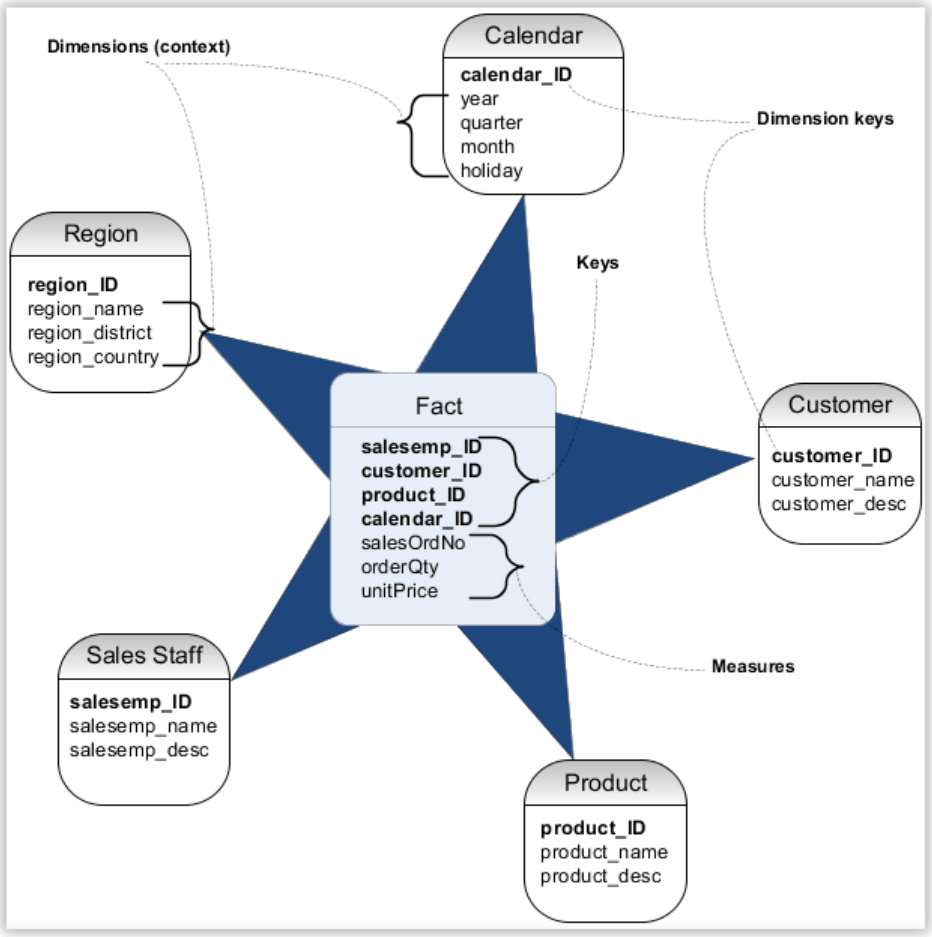


Figure 8 - Example of a star schema (Documentation Infocenter)

There is only a single fact table with three measures: "salesOrdNo", "orderQty", and "unitPrice". Each dimension communicated directly with the fact table.

2.7.2 Snowflake schema

The snowflake schema stores exactly the same data as the star schema. The fact table has the same dimensions as it does in the star schema example. The most important difference is that the dimension tables in the snowflake schema are normalized. Interestingly, the process of normalizing dimension tables is called snowflaking (Datawarehouse4u.Info).

Two main differences between both schemas are in terms of normalization and query complexity.

In terms of normalization we can find the following differences (Drkušić, 2016):

- Snowflake schemas use less space to store dimension tables. This is because as a rule any normalized database produces far fewer redundant records;
- Denormalized data model increases the chances of data integrity problems. These issues will complicate future changes and maintenance as well.

In terms of complexity, the snowflake schema query is more complex. Because the dimension tables are normalized, we need to dig deeper to get the name of the product type and the city. In fact, we have to add another JOIN for every new level in the same dimension. On the other hand, in the star schema, we only join the fact table with those dimension tables we need. At most, we'll have only one JOIN per dimension table.

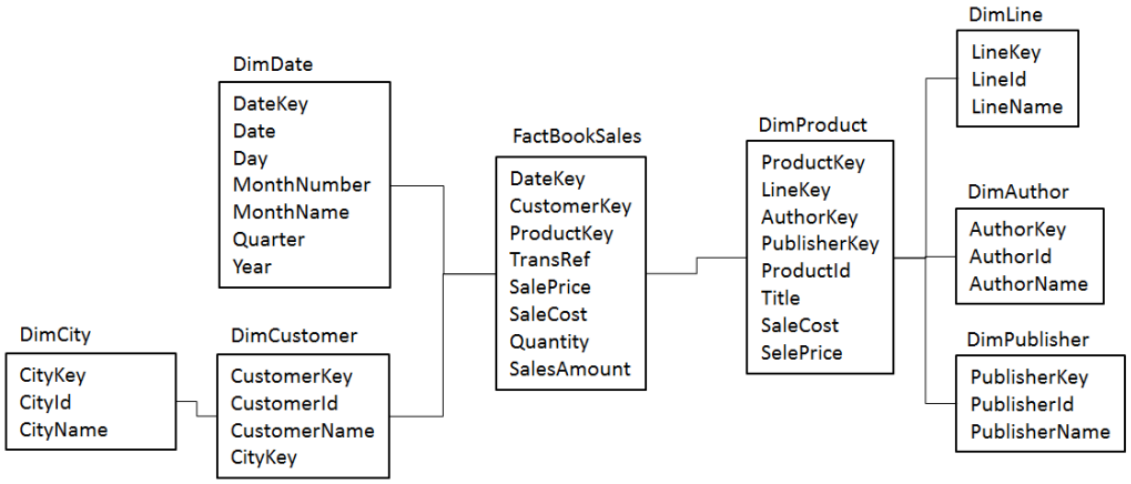


Figure 9 - Example of a snowflake schema (Rainardi, 2012)

There is also only a single fact table called "FactBookSales" and a total of 6 dimensions. Only three dimensions communicate directly with the fact table. For example the "DimCity" dimension is associated with the "DimCustomer" table.

Finally, Figure 10 provides a brief comparative analysis between the differences that may be found between the star and snowflake schema.

Comparison chart

	Snowflake Schema	Star Schema
Ease of maintenance / change	No redundancy and hence more easy to maintain and change	Has redundant data and hence less easy to maintain/change
Ease of Use	More complex queries and hence less easy to understand	Less complex queries and easy to understand
Query Performance	More foreign keys-and hence more query execution time	Less no. of foreign keys and hence lesser query execution time
Type of Datawarehouse	Good to use for datawarehouse core to simplify complex relationships (many:many)	Good for datamarts with simple relationships (1:1 or 1:many)
Joins	Higher number of Joins	Fewer Joins
Dimension table	It may have more than one dimension table for each dimension	Contains only single dimension table for each dimension
When to use	When dimension table is relatively big in size, snowflaking is better as it reduces space.	When dimension table contains less number of rows, we can go for Star schema.
Normalization/ De-Normalization	Dimension Tables are in Normalized form but Fact Table is still in De-Normalized form	Both Dimension and Fact Tables are in De-Normalized form
Data model	Bottom up approach	Top down approach

Figure 10 - Comparison between star and snowflake schemas (Diffen)

2.7.3 Constellation schema

For each star schema it is possible to construct fact constellation schema (for example by splitting the original star schema into more star schemas, each of them describes facts on another level of dimension hierarchies). The fact constellation architecture contains multiple fact tables that share many dimension tables.

The main shortcoming of the fact constellation schema is a more complicated design because many variants for particular kinds of aggregation must be considered and selected. Moreover, dimension tables are still large.

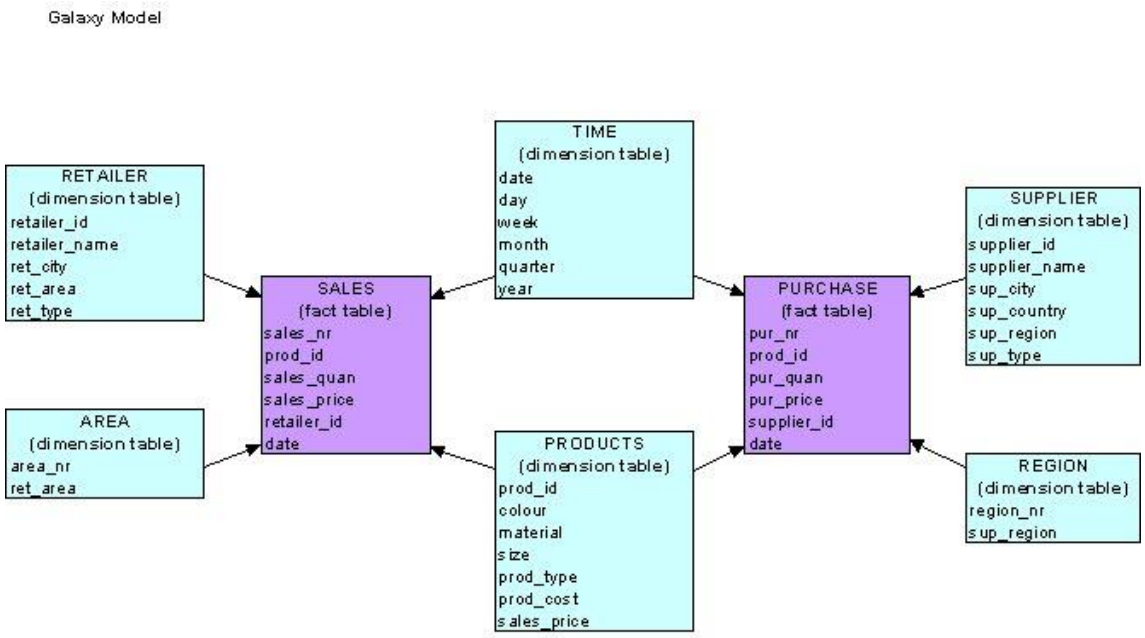


Figure 11 - Example of a constellation schema (Microsoft Technology)

There are two fact tables in the model: "Sales" and "Purchase". There are dimensions that are associated with both fact tables, and there are also dimensions that are only associated with one fact table. For example, the "Retailer" dimension can only be accessible by the "Sales" fact table.

2.8 Granularity

Granularity is one of the most important elements in the DW data modeling. It is important to understand the relationship between detailing and granularity. When we speak of lower granularity, or fine granularity, it means greater detailing (less summarization) of the data. Greater granularity, or gross granularity, means less detail (greater summarization). Therefore, we can realize that granularity and detailing are inversely proportional.

Granularity directly affects the volume of data stored, the speed of queries, and the level of detail of DW information. A greater detail level originates a greater level of flexibility to obtain answers. However, we will have a higher volume of data and slower speed of the queries. On the other hand, a smaller level of detail originates a smaller volume of data and a greater data summarization with better performance. However, the scope will be smaller, that is, the greater the restrictions on the information queries.

The balance between detailing and summarization must be evaluated so that granularity is modeled with the best efficiency and effectiveness for user queries, always taking into account the needs raised at the beginning of the project.

An example of applying the concept of granularity is given in Figure 12.

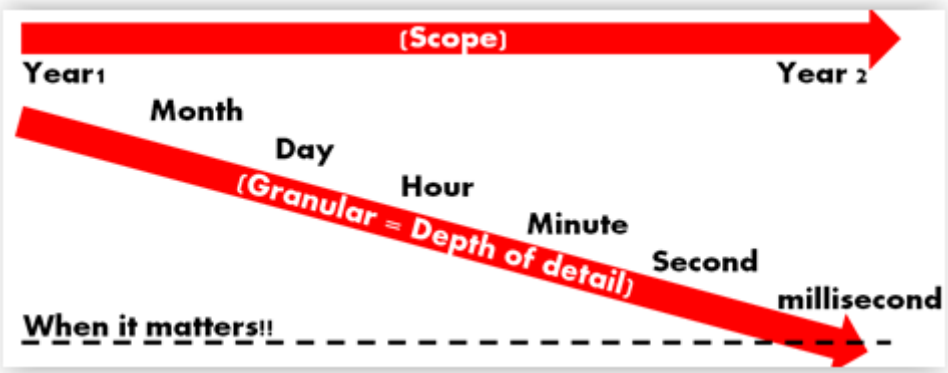


Figure 12 - Difference between scope and granularity

The granularity gives the level of detail. For example, for each year we can have detailed information regarding each month, day, hour, minute, second, and so on.

Let's consider another example, but this time looking for a bank scenario. In such scenario, it can exist three data levels, like it is shown in Figure 13.

Daily Detail	Monthly Summary	Quarterly Summary
Account	Account	Account
Activity Date	Month	Month
Amount	Number of transactions	Number of transactions
Deposit/Withdrawal	Withdrawals	Withdrawals
	Deposits	Deposits
	Beginning Balance	Beginning Balance
	Ending Balance	Ending Balance

Figure 13 - Data granularity in a bank scenario

2.9 ETL Process

ETL is a process for extracting data from a database system (DB), such data being processed, modified, and subsequently loaded into another BD. Studies report that ETL and data-cleaning tools consume one-third of the budget in a DW project and can, in terms of the development time of a DW project, consume 80% of that value.

Data warehouse projects consolidate data from different sources. Most of these sources tend to be relational databases or flat files, but there may be other types of sources as well. An ETL system needs to be able to communicate with databases and read various file formats used throughout the organization.

Figure 14 generally describes the ETL process. The bottom layer represents the storage of the data that is used throughout the process. At the Left-hand side, one can observe the "original" data coming from most Cases, of BD or, therefore, of files with heterogeneous formats, for

example of text. Data from these sources are obtained by extraction routines that provide information equal or modified, with respect to the original data source. Subsequently, these data are propagated to the Data Staging Area (DSA) where they are transformed and cleaned before being loaded into the DW. The DW is represented on the right of the figure and aims to store the data. Charging the data in the DW, is performed through the load activities represented in part upper right corner of the figure.

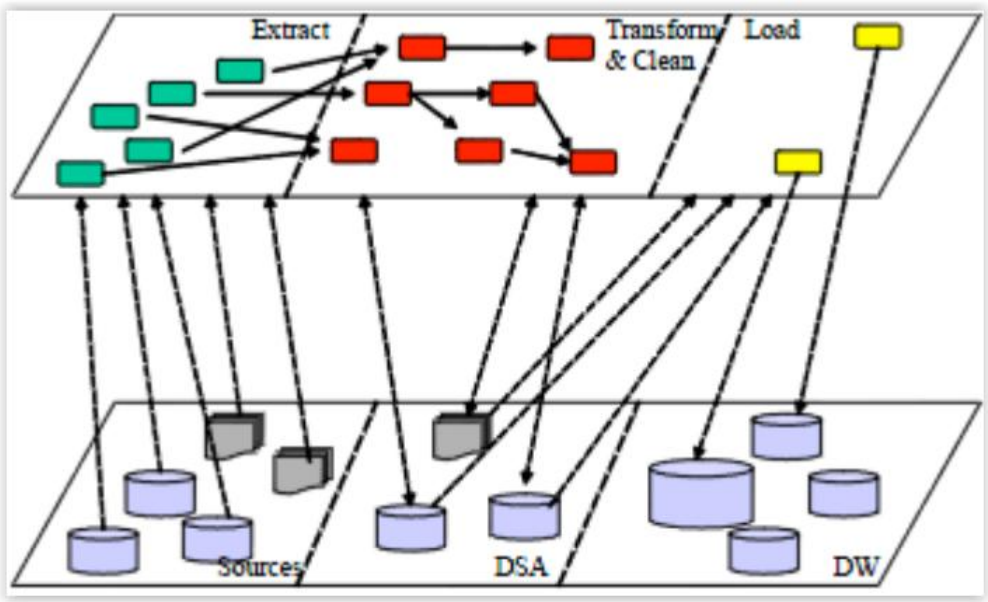


Figure 14 - Vision of ETL process

Before starting an ETL process, the following items must be properly aligned:

- Identification and clear vision of business requirements;
- Perform a feasibility analysis of the available data;
- Analysis the maximum time allowed for data latency;
- Apply the compliance and security policies adopted by the company.

The ETL process is generally composed of the following ten steps:

1) Determine all the target data needed in the DW;
2) Determine all the data sources, both internal and external;
3) Prepare data mapping for target data elements from sources;
4) Establish comprehensive data extraction rules;
5) Determine data transformation and cleansing rules;
6) Plan for aggregate tables;
7) Organize data staging area and test tools;
8) Write procedures for all data loads;
9) ETL for dimension tables;
10) ETL for fact tables.

2.9.1 Data extraction

The extraction activity of the source data includes data extraction routines that read the data, convert this data into an intermediate schema, and move it to an area, which is a temporary working are in which data are maintained in intermediate schemas. The data extraction process is subdivided into:

- Read data from the systems - the process of reading legacy system data can be a very simple or a very complex task. When the data source system is an open, well-documented legacy application, it will be an easy task. But as this is not usually the case, for the vast majority of the time, the origin of the data comes from legacy systems without documentation on the meaning of the data, and with an internal storage structure of complex understanding. However, there are worse scenarios, when the systems have a proprietary structure, where the format of the underlying files is not known;
- Determine changes to identify new data - identifying data to be loaded into the DW dramatically reduces the amount of data that will migrate to it. Distinguishing new data from previously read data in the extraction process can be a complex task. There are several techniques that are available to update a DW incrementally, a data mart or another operating system. Modified data capture techniques can fall into two general categories: static and incremental. Static data capture is usually associated with taking a snapshot of the data at a particular moment in time. In some cases, the complete set of data can be restored, but probably only a subset will be used. On the other side, increased data capture is a time-dependent model for capturing changes to operating systems. This technique is best applied in circumstances where data change is significantly less than the size of the data set for a specific period of time. These techniques are more complex than static capture, because they are linked to the DBMS or to the operating software that updates the DBMS. Three different techniques in this category can be applied: capture, trigger-based capture, and transaction log capture;
- Generalize key - a key management application must be adopted. Operational input switches usually need to be restructured before recorded. Very rarely an input key remains unchanged when read in the operating environment and written to the Data Warehouse environment. In simple cases, a time element is added to the key structure. In complex cases, the entire input key must go through a new hashing process, or be restructured;
- Combine records from multiple sources. In most DWs, the data come from several different and independent source systems. Establishing an intermediate data storage environment becomes necessary. To this end, the denormalization of dimensions data should be done in order to approach the final schema that will be loaded in the DW.

2.9.2 Data cleaning

Data cleansing is one of the most important processes in the ETL. The quality (perfection, validity, and accuracy) of Data Warehouse data should be dimensioned and informed, so that the decision makers can evaluate the reliability of the data, only then decide what measures to take. The other important question about data cleansing is knowing when to leave data

without cleanup. Legacy systems have a lot of dirty data, but because their functionality has not been affected because of this, there has never been a concern to clean up that data. Therefore, normally data that will migrate to the Data Warehouse environment requires correction and this implies a quality assessment of this data.

An overview of the data cleaning process is given in Figure 15. In general, data cleaning involves several phases:

- Data analysis;
- Definition of transformation workflow and mapping rules;
- Verification;
- Transformation;
- Backflow of cleaned data.

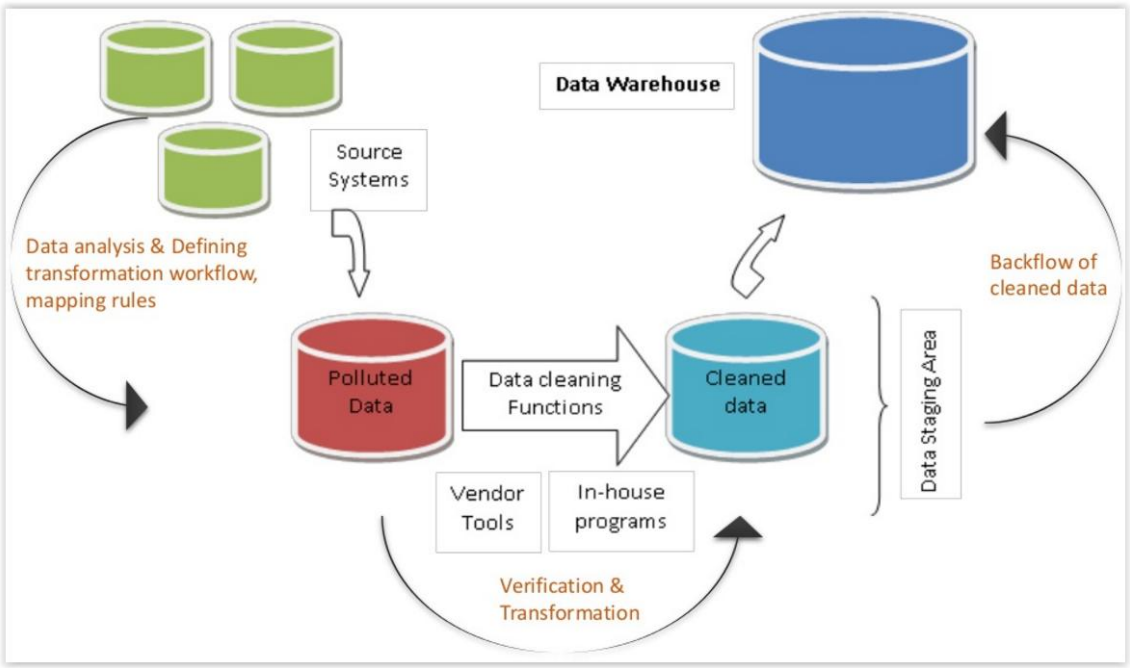


Figure 15 - Data cleaning process (Soni)

One of the hardest data cleansing issues is dealing with data that has multiple identifiers. When many files contain many redundant records about a single entity, and those have built-in meaning or keys not synchronously defined, one should seek to consolidate these duplicate records within a single file or database. To do this, a cross-reference table must be maintained to relate the record occurrence to records that previously existed, but no longer executes. This is used to redirect any business transaction that uses old identifiers to the record instance. An audit file with images of before and after data must also be maintained to ensure the reconstruction of the original records in case of deconsolidation. Files with a single record instance can be compared and consolidated by crossing redundant files that select the most reliable values per spread for the Data Warehouse. The data values of each source must be corrected and synchronized for consistency to a possible extent. The equivalence of the data

must be maintained in the redundant files, as well as the cross-reference table of the reported occurrences.

Data to be properly cleaned must be identified and cleaning strategies should be defined for each type of dirty data encountered. The most common types of dirty data are generated at the data entry stage, where some field is filled considering the importance of this information for the various sectors of the company. They can then be filled only to pass validity mechanisms of the frontend application, without guaranteeing the correctness of the value.

The most common types of dirt data are:

- Dummy values - are the values that the user normally needs to fill in the data entry application even without knowing the content of the information. The concern is only with the data type and any value fills the fields, so that pass on the validation review. On a client basis the most likely fields that can receive these amounts are social security number, customer age, postal code, or state information;
- Lack of data - fields in a file or a table may have missing data. One of the reasons for this is that departments of the same company have different needs for the existence of certain data in their operation. In this way, field filling may be required or not. For example, a department may require the capture of information that is totally dispensable;
- Multipurpose fields - a data field can be used for multiple purposes depends on the department that made the data entry and the cross reference in relation to other fields. In the first case, the Department for the purpose of meeting needs, resets the same data area several times. In the second case, the field to be observed will be filled according to the functionality of the business operation involved;
- Inconsistent data - Information entered in systems may have inconsistent content. For example, in a database of clients, the information about the city that the person lives does not match with the corresponding state information, that is, in the city field the information is Los Angeles and the state field of the content is Texas;
- Reuse of primary keys - this is one of the most critical issues about dirty data that can occur in legacy systems. Operating systems rarely store history beyond 90 or 180 days, which often causes primary key values to be reused. This situation will generate a huge problem of data integration in a Data Warehouse environment;
- Non-unique identifiers - the inclusion of multiple records for the same entity within the same system will pose a major data integration problem, since data that should be related will not be. For example, one customer identified by several different customer codes.

The data to be extracted from the legacy systems must be analyzed with the purpose of trying to identify all these types of dirty data described above. Adopted rules should be specified and documented as metadata. The data processing will be done with the aid of these transformation rules (metadata) defined for each case.

2.9.3 Data loading

After the data transformation is carried out the loading process, where the data is placed on the presentation server.

The destination of the data may be for atomic dart mart or aggregated Data Mart, each data target having its own details and syntax. The charging process is responsible for recognizing the differences to use them or avoiding them in the most appropriate way. In the same way, database techniques to optimize processes to avoid logging during the process can be invoked from databases or recorded in scripts.

There are commonly three strategies for data loading:

- Renewal - data previously archived are rewritten and then automatically updated;
- Logical or incremental update - it uses a non-destructive archive, where already archived data is added to other data;
- Physical update - it uses also a destructive archive, where the previously archived data is deleted and replaced entirely by the new data that will be loaded.

3. OLAP

3.1 The Concept

OLAP (Online Analytical Processing) is a software that enables business analysts, managers and executives to analyze and visualize business data quickly, consistently and primarily interactively. OLAP functionality is characterized by dynamic, multidimensional analysis of an organization's consolidated data, allowing end-user activities to be both analytical and navigational. OLAP tools are typically designed to work with denormalized databases. These tools are able to navigate data from a Data Warehouse, having a structure suitable for both research and presentation of information.

In recent years, the term Business Intelligence (BI) has been widely used in the market as synonymous with analytical systems, OLAP, cubes, among others. Although these denominations may be associated with each other, they are conceptually distinct. BI can be obtained by any artifact, whether technological or not, that allows the extraction of knowledge from business analysis. The effectiveness of these analyzes will be greater if the data are available in a consistent and preferably consolidated manner. Computerized BI solutions usually contain analytical systems, which can be of several types, depending on the purpose of the analyzes and the user profile.

3.2 Characteristics

Six essential characteristics can be seen in OLAP:

- Query-oriented technology - the main operation in OLAP environment is querying data;
- Data not changed - data is added to DW using the ETL process. Older data are not replaced by the new data. However, it can be migrated to a backup server;
- Data and queries are managed - it is important to guarantee a good performance of data stored in DW and also a good optimization process for the queries;
- Multidimensional data view - data are organized several dimensions of analysis;
- Complex calculations - math functions can be used to perform calculations on data;
- Time series - associated with data we have the notion of time.

There are significant differences between OLTP and OLAP systems. OLTP systems are designed for office workers while the OLAP systems are designed for decision makers. Besides that, other differences can be found and are synthesized in Figure 16.

OLTP vs. OLAP Differences

	OLTP	OLAP
Organization	By workflow per application	By dimension and business subject
Data Retention	Short term (2-6 months)	Long term (2-5 years)
Data Integration	Minimal or none	High, as part of ETL process
Data Storage	Gigabytes	Terabytes
Use	Real time Write & update Evenly distributed usage Transactional data	Batch load Reporting, read-only Spiked usage (based on time of warehouse loads)

Figure 16 - Differences between OLTP and OLAP (Darmawan, 2014)

3.3 Terminology

The OLAP basic terminology is composed of several elements (XLCubed, 2008): (i) cube; (ii) dimensions; (iii) measures; (iv) hierarchies; (v) member; and (vi) aggregation.

A cube is a data structure that aggregates measures by the levels and hierarchies of each of the dimensions. Cubes combine multiple dimensions (such as time, geography, and product lines) with summary data (such as sales or record numbers).

Dimensions are the business elements by which the data can be queried. They can be thought of as the 'by' part of reporting. For example "I want to see sales by region, by product by time". In this case region, product and time would be three dimensions within the cube, and sales would be a measure, below. A cube based environment allows the user to easily navigate and choose elements or combinations of elements within the dimensional structure.

Measures are the units of numerical interest, the values being reported on. Typical examples would be unit sales, sales value and cost.

A dimension can contain one or more hierarchies. Hierarchies are really navigable or drill paths through the dimension. They are structured like a family tree, and use some of the same naming conventions (children / parent / descendant). Hierarchies are what brings much of the power to OLAP reporting, because they allow the user to easily select data at different granularity (day / month / year), and to drill down through data to additional levels of detail. Figure 17 provides an example of the most common hierarchies that may be found in some given dimensions.

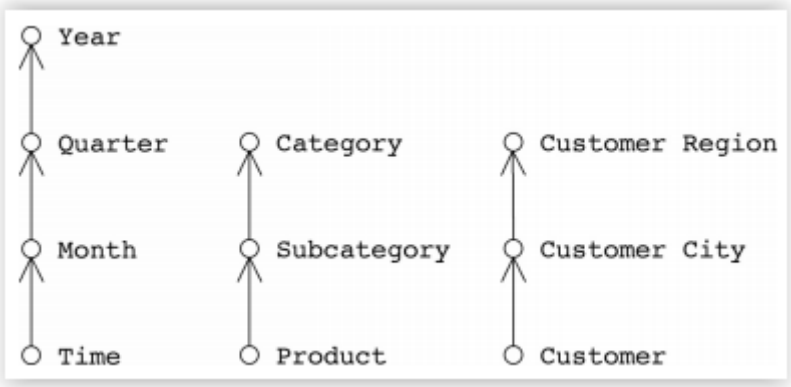


Figure 17 - Most common hierarchies (Rizzi, 2009)

A member is any single element within a hierarchy. For example, in a standard Time hierarchy, 1st January 2008 would be a member, as would 20th February 2008. However January 2008, or 2008 itself could also be members. The latter two would be aggregations of the days which belong to them. Members can be physical or calculated. Calculated members mean that common business calculations and metrics can be encapsulated into the cube, and are available for easy selection by the user, for example in the simplest case Profit = Sales - Cost.

Aggregation is a key part of the speed of cube based reporting. The reason why a cube can be very fast when for example selecting data for an entire year, is because it has already calculated the answer. Whereas a typical relational database would potentially sum millions of day level records on the fly to get an annual total, Analysis Services cubes calculate these aggregations during the cube build and hence a well designed cube can return the answer quickly. Sum is the most common aggregation method, but it's also possible to use average, max etc. For example, if storing dates as measures it makes no sense to sum them.

The cube introduces a number of dimensions, hierarchies and measures, modeling the business of interest, and all of which are available to the end user to quickly and easily select, drill, and slice and dice. With a well designed cube the user benefits from a reporting environment which is highly flexible, contains the pre-calculated business metrics they regularly use, and is fast in terms of data retrieval.

A data cube can be represented in a 2-D table, 3-D table or in a 3-D data cube. Let's consider a scenario where a company intends to keep track of sales, considering the dimensions: item, time, branch, and location. A possible representation of this information in a 2-D table is given in Figure 18.

Location="New Delhi"				
Time(quarter)	Item(type)			
	Entertainment	Keyboard	Mobile	Locks
Q1	500	700	10	300
Q2	769	765	30	476
Q3	987	489	18	659
Q4	666	976	40	539

Figure 18 - 2-D table representation of a data cube (Tutorials Point)

But here in this 2-D table, we have records with respect to time and item only. The sales for New Delhi are shown with respect to time, and item dimensions according to the type of items sold. If we want to view the sales data with one more dimension, say, the location dimension, then the 3-D view would be useful. The 3-D view of the sales data with respect to time, item, and location is shown in Figure 19.

Time	Location="Gurgaon"			Location="New Delhi"			Location="Mumbai"		
	Item			Item			Item		
	Mouse	Mobile	Modem	Mouse	Mobile	Modem	Mouse	Mobile	Modem
Q1	788	987	765	786	85	987	986	567	875
Q2	678	654	987	659	786	436	980	876	908
Q3	899	875	190	983	909	237	987	100	1089
Q4	787	969	908	537	567	836	837	926	987

Figure 19 - 3-D table representation of a data cube (Tutorials Point)

However, the 3-D table can be represented as a 3-D data cube as shown in the Figure 20.

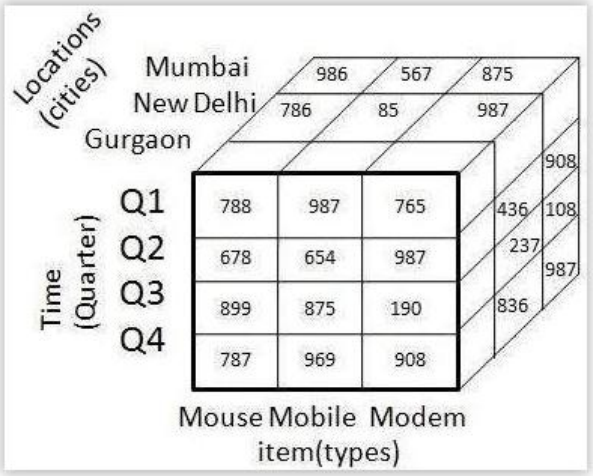


Figure 20 - 3-D data cube representation (Tutorials Point)

3.4 Operations

OLAP offers a wide range of operations. The most common operations are:

- Drill-down - disaggregates a dimension;
- Drill-across - involve more than one fact table and go down in the hierarchy;
- Roll-up - adds a dimension by going up in hierarchy;
- Drill-through - details beyond the cube. Goes to the records level;
- Slice - restricts a value across a dimension;
- Dice - it performs restrictions of values in several dimensions. It is applied to the values of the cells;
- Pivot - switches the view axis;
- Rank - sorts the members of a dimension according to some criteria;
- Rotate - performs a rotation of the dimension axes in a given direction;
- Split - performs permutation of values;
- Nest/Unest - reducing of dimensions;
- Push/Pull - merge values

Besides those most common operations, we can also use standard SQL operations such as junctions, unions, intersections and differences.

Below we detail how to use the most common operations in OLAP. For that, we use the example depicted in Figure 21. We register the number of sales and we consider the existence of four countries, two products and two years.

Country	Product	Year	No. of sales
Portugal	AAA	2017	25
Portugal	BBB	2017	45
Spain	AAA	2016	50
Spain	BBB	2016	65
Spain	AAA	2017	120
Spain	BBB	2017	80
South Korea	AAA	2017	170
Brazil	AAA	2016	60
Brazil	AAA	2017	75

Figure 21 - Example for OLAP operations

3.4.1 Drill-down

Drill-down refers to the process of viewing data at a level of increased detail. Considering our example, we will perform a drill-down (country). In Figure 22, we can see the inclusion of a new dimension called "city". The total of sales is 45 for the product "BBB" in Portugal for the year of 2017. For that we need to sum the number of sales for the city of Porto and Lisbon. A similar situation happens in the other rows of the table.

Country	City	Product	Year	No. of sales
Portugal	Porto	AAA	2017	25
Portugal	Porto	BBB	2017	35
Portugal	Lisbon	BBB	2017	10
Spain	Barcelona	AAA	2016	20
Spain	Madrid	AAA	2016	20
Spain	Valencia	AAA	2016	10
Spain	Madrid	BBB	2016	65
Spain	Madrid	AAA	2017	120
Spain	Madrid	BBB	2017	80
South Korea	Seoul	AAA	2017	170
Brazil	Recife	AAA	2016	50
Brazil	Manaus	AAA	2016	10
Brazil	Brazil	AAA	2017	75

Figure 22 - Drill-down operation

3.4.2 Roll-up

Roll-up refers to the process of viewing data with decreasing detail. Considering our example, we will perform a roll-up (country). Several situations can happen, but we will consider two scenarios. In the first scenario we have the information regarding the continent that belongs each country. The result for such scenario is presented in Figure 23. It appears four lines for the "Europe" continent, because there are two products and two years.

Continent	Product	Year	No. of sales
Europe	AAA	2017	145
Europe	BBB	2017	125
Europe	AAA	2016	50
Europe	BBB	2016	65
Asia	AAA	2017	170
South America	AAA	2016	60
South America	AAA	2017	75

Figure 23 - Roll-up operation (scenario I)

In the second scenario we consider that the "country" dimension will disappear. The data are aggregated considering this dimension. The result is depicted in Figure 24. There are only four lines in the table with information regarding each product and year.

Product	Year	No. of sales
AAA	2017	390
BBB	2017	125
AAA	2016	110
BBB	2016	65

Figure 24 - Roll-up operation (scenario I)

3.4.3 Slice and dice

Slice and dice refer to a strategy for segmenting, viewing and understanding data in a database. Users slices and dice by cutting a large segment of data into smaller parts, and repeating this process until arriving at the right level of detail for analysis. Slicing and dicing helps provide a closer view of data for analysis and presents data in new and diverse perspectives.

First, we will explain the use of slice considering our example. Let's make a slide per year like it is shown in Figure 25 (e.g., slice(Year="2017")). All information regarding the year of 2016 was omitted.

Country	Product	Year	No. of sales
Portugal	AAA	2017	25
Portugal	BBB	2017	45
Spain	AAA	2017	120
Spain	BBB	2017	80
South Korea	AAA	2017	170
Brazil	AAA	2017	75

Figure 25 - Slice operation

Then, we will use the dice operation that has a very similar function, but it uses more than one dimension. Let's consider a dice per year and number of sales. The operation will be the

following: dice(Year="2017" and No. of sales > 100). The result of this operation is depicted in Figure 26. Only records that have both conditions appeared in the result.

Country	Product	Year	No. of sales
Spain	AAA	2017	120
South Korea	AAA	2017	170

Figure 26 - Dice operation

3.4.4 Pivoting

Pivoting doesn't make any effect on the data but changes the way how dimensions are shown. Considering our example, we will perform a pivoting per year. Therefore the "year" dimension will be our first column in our table, as shown in Figure 27.

Year	Country	Product	No. of sales
2016	Spain	AAA	50
2016	Spain	BBB	65
2016	Brazil	AAA	60
2017	Portugal	AAA	25
2017	Portugal	BBB	45
2017	Spain	AAA	120
2017	Spain	BBB	80
2017	South Korea	AAA	170
2017	Brazil	AAA	75

Figure 27 - Pivoting operation

3.4.5 Rank

The "rank" operation is another function that doesn't perform any change in the data. However, the elements are ordered by a given dimension. Let's consider the following function: rank(No. of sales). The result of such operation is depicted in Figure 28.

Country	Product	Year	No. of sales
Portugal	AAA	2017	25
Portugal	BBB	2017	45
Spain	AAA	2016	50
Brazil	AAA	2016	60
Spain	BBB	2016	65
Brazil	AAA	2017	75
Spain	BBB	2017	80
Spain	AAA	2017	120
South Korea	AAA	2017	170

Figure 28 - Rank operation

3.5 Architecture

The most common architectures for OLAP are: (i) ROLAP; (ii) MOLAP; and (iii) HOLAP. A comparative analysis among these architectures are given in Figure 29.

Architecture	Performance	Scalability	Cost
MOLAP	High	Low	High
ROLAP	Low	High	Low
HOLAP	High	High	High

Figure 29 - Comparative analysis of MOLAP, ROLAP and HOLAP architectures

There are also other architectures not so common that appears in OLAP. Among them, we highlight the DOLAP, JOLAP and SOLAP.

The DOLAP architecture is an OLAP desktop architecture, that is, it is a tool for users who have a copy of the multidimensional database or a subset of it, or who want to access a central data repository locally. The user accesses this repository, triggers an SQL statement and accesses the existing cubes in the multidimensional database residing on the OLAP server and returns one to be analyzed on its workstation. The advantage of this architecture is to reduce the overhead on the database server since all OLAP processing happens on the client machine and the disadvantage is the size of the micro-cube that cannot be very large, otherwise the analysis can be time-consuming and client doesn't support it.

JOLAP is a Java API for OLAP, and SOLAP is the application of OLAP for geographic information systems.

3.5.1 MOLAP

In the MOLAP architecture the data is stored in a multidimensional database, where the MOLAP server operates and the user works, mounts and manipulates the different data on the server. Data from a multidimensional database is stored in a space smaller than that used to store the same data in a relational database. In the multidimensional database, data are kept in array data structures in order to provide better performance when accessing them. In addition to being a fast architecture another advantage is the rich and complex set of analysis functions present in multidimensional databases.

One of its limitations is the possibility of the data being sparse (not all crossing the dimensions contains data), occurring the so-called data storage explosion, that is, a huge multidimensional database containing little data stored. Other limitations of this tool are related to the fact that multidimensional banks are proprietary systems that do not follow standards, that is, each developer creates his own structure for the bank and the support tools themselves.

Figure 30 provides an example adopting the MOLAP architecture. Previous selected data are loaded in MOLAP server. SQL language can also be used to process data, if needed. Calculations can be made using directly OLAP cubes.

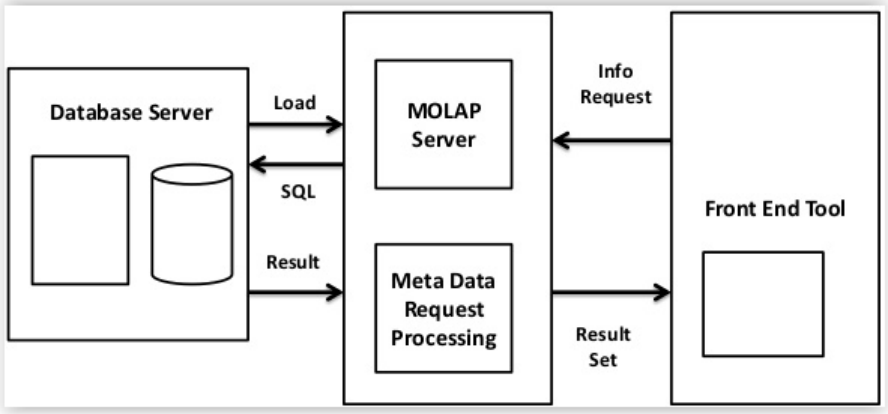


Figure 30 - MOLAP architecture

The main advantages of MOLAP include:

- High performance - cubes are built for fast data recovery;
- Can perform complex calculations - all calculations are pre-generated when the cube is created and can be easily applied at the time of the data search.

On the other side, the main disadvantages are:

- Low scalability - its advantage of achieving high performance with the pre-generation of all calculations at the time of cube creation makes MOLAP limited to a small amount of data. This deficiency can be circumvented by including only the summary of the calculations when constructing the cube;
- High investments: this model requires huge additional investments as a proprietary technology hub.

3.5.2 ROLAP

The ROLAP architecture is a simulation of OLAP technology made in relational databases that, by using the relational structure, has the advantage of not restricting the volume of data storage. This tool does not use pre-calculated cubes like MOLAP. As the user mounts his query in a graphical interface, the tool accesses the metadata or any other resources that it has, to generate an SQL query.

Its main feature is the possibility of making any query, better serving users who do not have a well defined analysis scope. This tool has the advantage of using established technology, open architecture and standardized, benefiting from the diversity of platforms, scalability and hardware parallelism. Its disadvantage is the poor set of functions for dimensional analysis and the poor performance of the SQL language in the execution of heavy queries.

Figure 31 provides an example of adopting a ROLAP architecture. SQL is used to process the data directly in the ROLAP server.

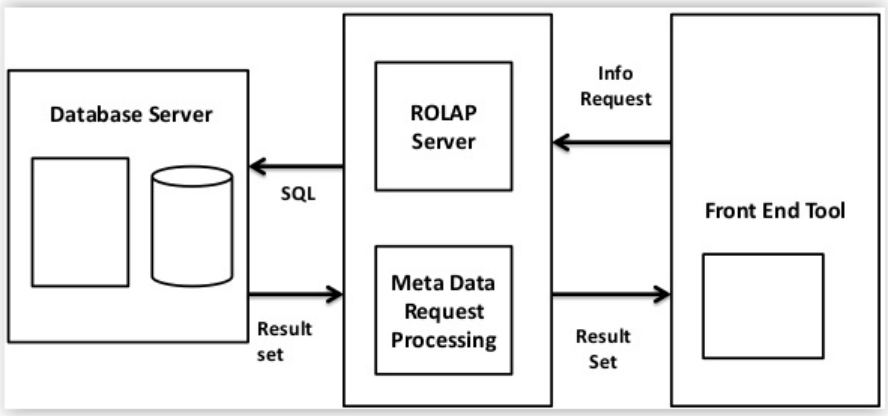


Figure 31 - ROLAP architecture

The main advantages of ROLAP include:

- High scalability - using the ROLAP architecture, there is no restriction on the quantity of data to be analyzed, being this limitation only in terms of the relational database used;
- Take advantage of the inherent functionality of the relational database - many relational databases already come with a number of features and the ROLAP architecture can leverage these features.

On the other side, the main disadvantages of ROLAP are:

- Low performance - each ROLAP report is basically an SQL query (or multiple SQL queries) in the relational database and a query can be significant time consuming if there is a large amount of data;
- Limited by SQL features: ROLAP relies primarily on generating SQL statements to query the relational database, but these statements do not meet all the requirements. For example, it is difficult to perform complex calculations using SQL.

3.5.3 HOLAP

HOLAP architecture, or hybrid processing, has become more popular for today's products because it can combine the capabilities and scalability of ROLAP tools with the superior performance of multidimensional databases. For example, assume a base of 50,000 customers in 300 cities, 20 states, 5 regions and a grand total. Up to the cities level multidimensional storage would resolve queries to raise sales totals. However, if it were necessary to query a customer's total sales, the relational database would respond much faster to the request. This situation is typical for indicating the HOLAP architecture.

Figure 32 provides an example adopting the HOLAP architecture. In this scenario, MOLAP server and relational data servers can co-exist. Both result sets are processed to frontend tool.

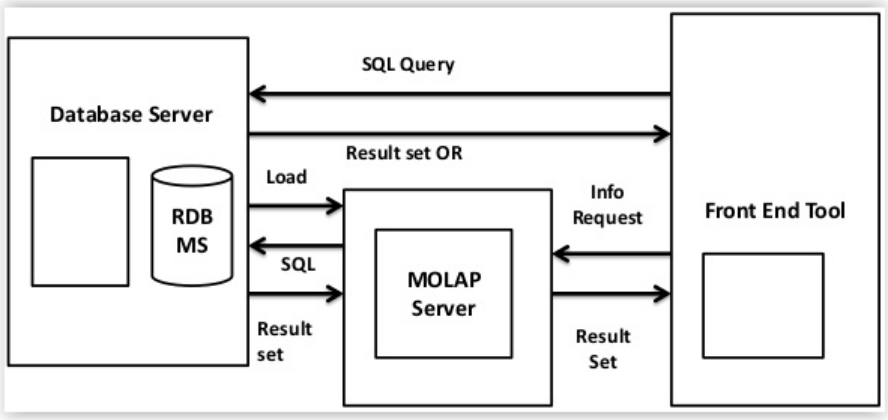


Figure 32 - HOLAP architecture

The main advantages of HOLAP include:

- High performance - dimensional cubes only store information synthesis;
- High scalability - the details of the information are stored in a relational database.

On the other side, the main disadvantages are:

- Complex architecture - this model presents the highest acquisition and maintenance costs.

3.6 Virtual Cubes

A virtual cube is a logical view of parts of one or more cubes, in which dimensions and measurements are selected from the original cubes and included in the virtual cube.

Virtual cubes are often likened to views in a relational database. A virtual cube merges portions of two existing cubes so that a combination of dimensions and measures can be analyzed through the single, virtual cube. For example, a retailer may have two cubes: one that stores the number of visitors to its website and another that stores purchases. A virtual cube could be used to correlate the data from both cubes to calculate the average sales per website visit (Search Data Management).

Virtual cubes can also be used to prevent unauthorized users from viewing private or sensitive information. For example, if a cube has both sensitive and non-sensitive information, the non-sensitive information can be made available in a virtual cube for those users who need it. The sensitive data, meanwhile, remain in the existing cube where it is accessed by authorized users.

Virtual cubes offer the following benefits:

- Storage and performance can be optimized on a case-by-case basis. In this way, it becomes possible to maintain the best design approach for each individual cube;

- Allows the possibility of having overall analysis, keeping for the sake of simplicity, the separate cubes. In this sense, users can query cubes together as long as they share at least one common dimension.

3.7 Partitioning

Partitioning is done to improve performance and make data management easier. Partitioning also helps balance the various system requirements. It optimizes hardware performance and simplifies data warehouse management by dividing each fact table into several separate partitions.

Partitioning can be done for the following reasons (Tutorials Point):

- For easy management - the fact table in a data warehouse can grow up to hundreds of gigabytes in size. This huge size of the fact table is very hard to manage as a single entity. Therefore, it needs partitioning;
- To assist backup/recovery - If we do not partition the fact table, then we have to load the complete fact table with all the data. Partitioning allows us to load only as much data as is required on a regular basis. It reduces the time to load and also enhances the performance of the system;
- To enhance performance - by partitioning the fact table into sets of data, the query procedures can be enhanced. Query performance is enhanced because now the query scans only those partitions that are relevant. It does not have to scan the whole data.

There are generally three types of partitioning: (i) horizontal; (ii) vertical; and (iii) hardware.

In the horizontal partitioning the fact table is partitioned after the first few thousand entries. This is because in most cases, not all the information in the fact table needed all the time. Therefore, horizontal partitioning helps to reduce the query access time, by directly cutting down the amount of data to be scanned by the queries. Horizontal partitioning the fact table is a good way to seep up queries, by mining the set of data to be scanned (without using an index). Different strategies can be used for horizontal partitioning. Among them we highlight:

- Partitioning by time, which typically conduced to different sized segments;
- Partitioning by geographical location, which typically conduces very asymmetric sized segments;
- Partitioning by size of table, which typically implies that are tables that will never be partitioned;
- Using round robin partitions, which is typically more difficult to manage.

Vertical partitioning, splits the data vertically. Vertical partitioning can be performed using a normalization or a row splitting technique. The Figure 33 depicts how vertical partitioning is done.

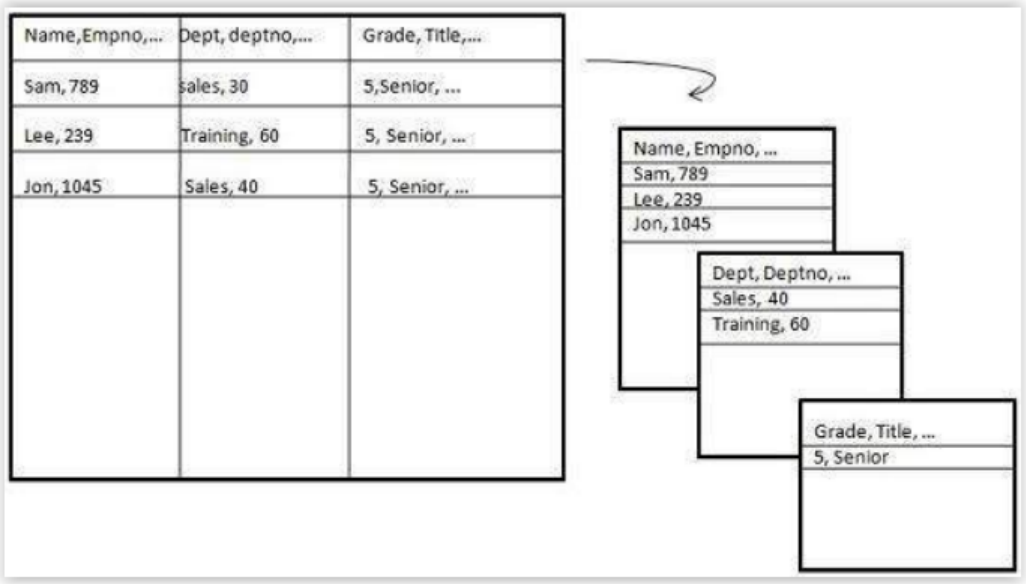


Figure 33 - Vertical partitioning approach

The usual approach in normalization in database applications is to ensure that the data is divided into two or more tables, such that when the data in one of them is updated, it does not lead to anomalies of data. However, normalizing data in a DW can lead to large, inefficient join operations.

The row splitting method involves identifying the not so frequently used field and putting them into another table. This would ensure that the frequently used field can be accessed more often, at a much lesser computation time. A good idea is to consider row splitting a fact table if some columns are accessed infrequently.

Another possibility is to use hardware partitioning. The idea is to optimize the database by respecting the specific hardware architecture. The exact details of optimization depend on the hardware platforms. However, some guidelines can be used:

- Maximize the processing power availability;
- Minimize disk accessed and I/O operations;
- Reduce bottlenecks at the CPU and I/O throughput.

Bibliography

1keydata. (n.d.). *Data Warehouse Definition*. Retrieved 08 11, 2017, from http://www.1keydata.com/datawarehousing/data-warehouse-definition.html

Algolytics. (n.d.). *Analytical Data Marts – data analyst's indispensable tool*. Retrieved 08 13, 2017, from http://algolytics.com/analytical-data-marts-data-analysts-indispensable-tool/

Business Dictionary. (n.d.). *Data Warehouse*. Retrieved 08 11, 2017, from http://www.businessdictionary.com/definition/data-warehouse.html

Business Impact. (2015, 05 07). *The Pros & Cons of Data Warehouses*. Retrieved from http://businessimpactinc.com/blog/the-pros-cons-of-data-warehouses/

Darmawan, N. (2014, 01 03). *Business Intelligence - OLTP vs OLAP (Differences)*. Retrieved from http://www.nicobudidarmawan.com/2014/01/business-intelligence-oltp-vs-olap.html

Datawarehouse4u.Info. (n.d.). *Data Warehouse*. Retrieved 08 13, 2017, from http://datawarehouse4u.info/index_en.html

Diffen. (n.d.). *Snowflake Schema vs. Star Schema*. Retrieved 08 13, 2017, from http://www.diffen.com/difference/Snowflake_Schema_vs_Star_Schema

Documentation Infocenter. (n.d.). *Star Schema*. Retrieved 08 13, 2017, from https://docs.infor.com/help_lawson_cloudsuite_10.0/index.jsp?topic=%2Fcom.lawson.help.reporting%2Fcom.lawson.help.bpwag-w_10.4.0%2FL55461185818015.html

Drkušić, E. (2016, 04 28). *Star Schema vs. Snowflake Schema*. Retrieved from http://www.vertabelo.com/blog/technical-articles/data-warehouse-modeling-star-schema-vs-snowflake-schema

Informatica. (n.d.). *What is Data Warehousing?* Retrieved 08 11, 2017, from https://www.informatica.com/services-and-training/glossary-of-terms/data-warehousing-definition.html#fbid=UxdjAEPUMd3

Kumar, A. (2012, 04 08). *Concept 5: Data Mart Vs Data Warehouse*. Retrieved from https://learnibm.wordpress.com/category/datawarehouse-concepts/page/2/

Microsoft Technology. (n.d.). *BI: Dimensional Model - Fact Constellation schema architecture*. Retrieved from http://blog-mstechnology.blogspot.pt/2010/06/bi-dimensional-model-fact-constellation.html

Mitschang, B. (n.d.). *Data-Warehouse-, Data-Mining- und OLAP-Technologien*. Retrieved 08 19, 2017, from https://www.ipvs.uni-stuttgart.de/export/sites/default/ipvs/abteilungen/as/lehre/lehrveranstaltungen/vorlesungen/WS1415/material/chapter02.pdf

Mullins, C. (n.d.). *The benefits of deploying a data warehouse platform*. Retrieved 08 13, 2017, from Search Data Management: http://searchdatamanagement.techtarget.com/feature/The-benefits-of-deploying-a-data-warehouse-platform

Ponniah, P. (2001). *DATA WAREHOUSING FUNDAMENTALS.* New York: John Wiley & Sons.

Power, D. (2008, 12 03). *What are advantages and disadvantages of data warehouses?* Retrieved from http://dssresources.com/faq/index.php?action=artikel&id=180

Rainardi, V. (2012, 06 16). *The Main Weakness of Snowflake Schemas*. Retrieved from https://dwbi1.wordpress.com/2012/07/16/the-main-weakness-of-snowflake-schemas/

Rea, A. (n.d.). *Data Mining*. Retrieved 08 11, 2017, from The Queen's University of Belfast: http://www.pcc.qub.ac.uk/tec/courses/datamining/stu_notes/dm_book_2.html

Rizzi, G. &. (2009, 04 21). *Data Warehouse Design: Modern Principles and Methodologies*. Retrieved from http://cdn.ttgtmedia.com/searchDataManagement/downloads/Data_Warehouse_Design.pdf

Saranya, V. (n.d.). *OLAP*. Retrieved 08 19, 2017, from https://www.slideshare.net/ersaranya/olap-27655941

Search Data Management. (n.d.). Retrieved 08 19, 2017, from http://searchdatamanagement.techtarget.com/definition/virtual-cube

Soni, R. (n.d.). *Role of the data cleaning in Data Warehouse*. Retrieved 08 14, 2017, from https://www.slideshare.net/ramakantsoni/role-of-data-cleaning-rk

Thakur, S. (2016, 06). *9 Disadvantages and Limitations of Data Warehouse*. Retrieved from http://whatisdbms.com/9-disadvantages-and-limitations-of-data-warehouse/

Tutorials Point. (n.d.). *Data Warehousing - Terminologies*. Retrieved 08 19, 2017, from https://www.tutorialspoint.com/dwh/dwh_terminologies.htm

XLCubed. (2008, 11 26). *THE BASIC STRUCURE OF A CUBE*. Retrieved from https://blog.xlcubed.com/2008/11/the-basic-strucure-of-a-cube/