|  |
| --- |
| EPAM Systems, RD Dep. |
| MTN.BI.07 Data Warehouse Architecture |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status | [Kiryl Bucha](mailto:Kiryl_Bucha@epam.com) | 12-JAN-2012 |  |  |
| 2.0 | Updated in accordance with renewed content | [Elias Nema](mailto:Elias_Nema@epam.com) | 20-JAN-2014 |  |  |

Contents

[1. Data Warehouse Architectures 3](#_Toc383290519)

[1.1. Single-Layer Architecture 3](#_Toc383290520)

[1.2. Two-Layer Architecture 3](#_Toc383290521)

[1.3. Three-Layer Architecture 4](#_Toc383290522)

[1.4. An Additional Architecture Classification 5](#_Toc383290523)

[1.5. Physical Model 8](#_Toc383290524)

[1.5.1. Staging layer 8](#_Toc383290525)

[1.5.2. Foundation layer - Third Normal Form 10](#_Toc383290526)

[1.5.3. Access layer - Star Schema 10](#_Toc383290527)

[1.6. Meta-data 10](#_Toc383290528)

[2. Accessing Data Warehouses 10](#_Toc383290529)

[2.1. Reports 10](#_Toc383290530)

[2.2. OLAP 11](#_Toc383290531)

[2.3. Dashboards 11](#_Toc383290532)

[3. OLAP, MOLAP, and HOLAP 11](#_Toc383290533)

[4. Additional Issues 12](#_Toc383290534)

[4.1. Quality 12](#_Toc383290535)

[4.2. Security 13](#_Toc383290536)

[4.3. Evolution 13](#_Toc383290537)

[5. The Interim Results 14](#_Toc383290538)

[6. Source Books and Articles 16](#_Toc383290539)

# Data Warehouse Architectures

The following architecture properties are essential for a data warehouse system:

* Separation. Analytical and transactional processing should be kept apart as much as possible.
* Scalability. Hardware and software architectures should be easy to upgrade as the data volume, which has to be managed and processed, and the number of users’ requirements, which have to be met, progressively increase.
* Extensibility. The architecture should be able to host new applications and technologies without redesigning the whole system.
* Security. Monitoring accesses is essential because of the strategic data stored in data warehouses.
* Administerability. Data warehouse management should not be overly difficult.

Two different classifications are commonly adopted for data warehouse architectures. Separation is a structure-oriented one that depends on the number of layers used by the architecture. Scalability depends on how the different layers are employed to create enterprise-oriented or department-oriented views of data warehouses.

## Single-Layer Architecture

A single-layer architecture is not frequently used in practice. Its goal is to minimize the amount of data stored; to reach this goal, it removes data redundancies. In this case, data warehouses are virtual.

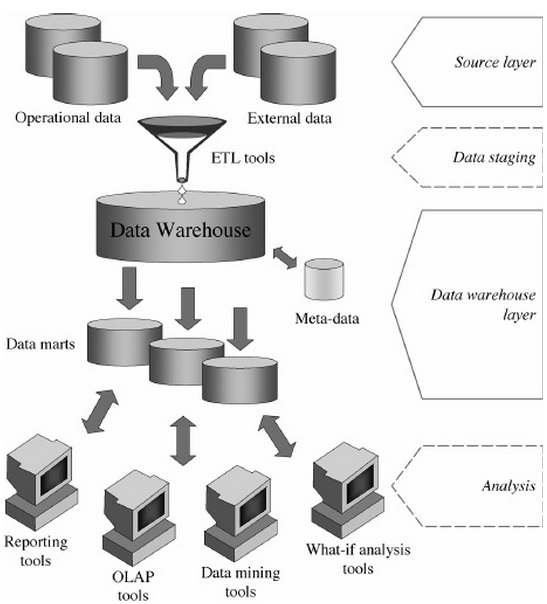
The weakness of this architecture lies in its failure to meet the requirement for separation between analytical and transactional processing. Analysis queries are submitted to operational data after the middleware interprets them. It this way, the queries affect regular transactional workloads. In addition, although this architecture can meet the requirement for integration and correctness of data, it cannot log more data than sources do. For these reasons, a virtual approach to data warehouses can be successful only if analysis needs are particularly restricted and the data volume to analyze is huge.

## Two-Layer Architecture

The requirement for separation plays a fundamental role in defining the typical architecture for a data warehouse system, as shown in Figure 1. Although it is typically called a two-layer architecture to highlight a separation between physically available sources and data warehouses, it actually consists of five subsequent data flow stages:

1. **Source layer.** A data warehouse system uses heterogeneous sources of data. That data is originally stored to corporate relational databases or legacy databases, or it may come from information systems outside the corporate walls.
2. **Data staging.** The data stored to sources should be extracted, cleansed to remove inconsistencies and fill gaps, and integrated to merge heterogeneous sources into one common schema. The so-called Extraction, Transformation, and Loading tools (ETL) can merge heterogeneous schemata, extract, transform, cleanse, validate, filter, and load source data into a data warehouse. You must clean and process your operational data before putting it into the warehouse. You can do this programmatically, although most data warehouses use a staging area instead. A staging area simplifies building summaries and general warehouse management.
3. **Data warehouse layer.** Information is stored to one logically centralized single repository: a data warehouse. The data warehouse can be directly accessed, but it can also be used as a source for creating data marts. Meta-data repositories store information on sources, access procedures, data staging, users, data mart schemata, and so on.
4. **Data Marts.** You may want to customize your warehouse's architecture for different groups within your organization. You can do this by adding data marts, which are systems designed for a particular line of business.
5. **Analysis.** In this layer, integrated data is efficiently and flexibly accessed to issue reports, dynamically analyze information, and simulate hypothetical business scenarios. Technologically speaking, it should feature aggregate data navigators, complex query optimizers, and user-friendly GUIs.

The architectural difference between data warehouses and data marts needs to be studied closer. The component marked as a data warehouse in Figure 1 is also often called the primary data warehouse or corporate data warehouse. It acts as a centralized storage system for all the data being summed up. Data marts can be viewed as small, local data warehouses replicating (and summing up as much as possible) the part of a primary data warehouse required for a specific application domain.



**Figure 1 Two Layer DWH Architecture**

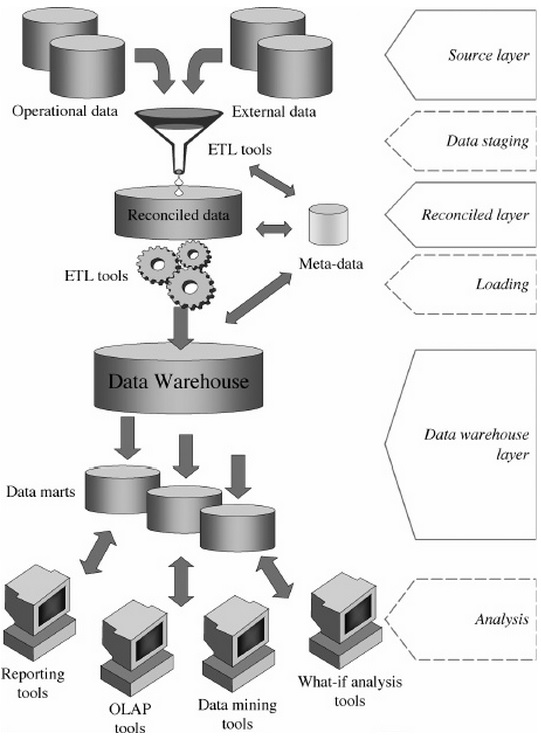
***Data Marts*** *Data marts are an important component of the front room. A data mart is a set of dimensional tables supporting a business process.*

* Data marts are based on the source of data, not on a department’s view of data. In other words, there is only one orders data mart in a product-oriented company. All the end user query tools and applications in various departments access this data mart to have a single, consistently labeled version of orders.
* Data marts contain all atomic detail needed to support drilling down to the lowest level. The view that data marts consist only of aggregated data is one of the most fundamental mistakes a data warehouse designer can make. Aggregated data in the absence of the lowest-level atomic data presupposes the business question and makes drilling down impossible. We will see that a data mart should consist of a continuous pyramid of identically structured dimensional tables, always beginning with the atomic data as the foundation.
* Data marts can be centrally controlled or decentralized. In other words, an enterprise data warehouse can be physically centralized on a single machine and the deployment of data marts can wait until a certain level of integration takes place in the ETL staging areas, or the data marts can be developed separately and asynchronously while at the same time participating in the enterprise’s conformed dimensions and facts. We believe that the extreme of a fully centralized and fully prebuilt data warehouse is an ideal that is interesting to talk about but is not realistic. A much more realistic scenario is the incrementally developed and partially decentralized data warehouse environment. After all, organizations are constantly changing, acquiring new data sources, and needing new perspectives. So in a real environment, we must focus on incremental and adaptable strategies for building data warehouses, rather than on idealistic visions of controlling all information before a data warehouse is implemented.

## Three-Layer Architecture

In this architecture, the third layer is the reconciled data layer or operational data store. This layer materializes operational data obtained after integrating and cleansing source data. As a result, those data are integrated, consistent, correct, current, and detailed. Figure 2 shows a data warehouse that is not populated from its sources directly, but from reconciled data.

The main advantage of the reconciled data layer is that it creates a common reference data model for a whole enterprise. At the same time, it sharply separates the problems of source data extraction and integration from those of data warehouse population. Remarkably, in some cases, the reconciled layer is also directly used to better accomplish some operational tasks, such as producing daily reports that cannot be satisfactorily prepared using the corporate applications, or generating data flows to feed external processes periodically so as to benefit from cleaning and integration. However, reconciled data leads to more redundancy of operational source data. Note that we may assume that even two-layer architectures can have a reconciled layer that is not specifically materialized, but only virtual, because it is defined as a consistent integrated view of operational source data.

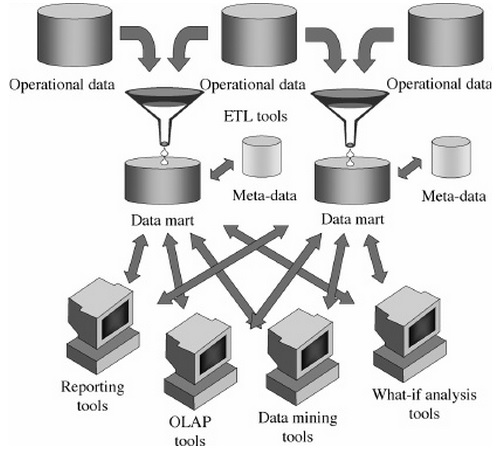


**Figure 2 Three Layer DWH Architecture**

## An Additional Architecture Classification

The scientific literature often distinguishes five types of architecture for data warehouse systems, in which the same basic layers mentioned in the preceding paragraphs are combined in different ways.

In independent data marts architecture, different data marts are separately designed and built in a nonintegrated fashion. This architecture can be initially adopted in the absence of a strong sponsorship toward an enterprise-wide warehousing project, or when the organizational divisions that make up the company are loosely coupled. However, it tends to be soon replaced by other architectures that better achieve data integration and cross-reporting.

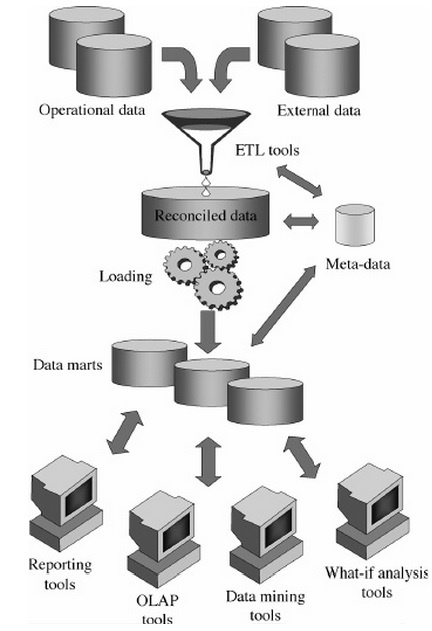


**Figure 3 Independent Data Mart Architecture**

The bus architecture, recommended by Ralph Kimball, is apparently similar to the preceding architecture, with one important difference. A basic set of conformed dimensions (that is, analysis dimensions that preserve the same meaning throughout all the facts they belong to), derived by a careful analysis of the main enterprise processes, is adopted and shared as a common design guideline. This ensures logical integration of data marts and an enterprise-wide view of information.

In the hub-and-spoke architecture, one of the most used in medium to large contexts, there is much attention to scalability and extensibility, and to achieving an enterprise-wide view of information. Atomic, normalized data is stored in a reconciled layer that feeds a set of data marts containing summarized data in multidimensional form (Figure 4). Users mainly access the data marts, but they may occasionally query the reconciled layer.

The centralized architecture, recommended by Bill Inmon, can be seen as a particular implementation of the hub-and-spoke architecture, where the reconciled layer and the data marts are collapsed into a single physical repository.



**Figure 4 Hub-and-spoke Architecture**

The federated architecture is sometimes adopted in dynamic contexts where preexisting data warehouses/data marts are to be noninvasively integrated to provide a single, cross-organization decision support environment (for instance, in the case of mergers and acquisitions). Each data warehouse/data mart is either virtually or physically integrated with the others, leaning on a variety of advanced techniques such as distributed querying, ontologies, and meta-data interoperability.

The following list includes the factors that are particularly influential when it comes to choosing one of these architectures:

* The amount of interdependent information exchanged between organizational units in an enterprise and the organizational role played by the data warehouse project sponsor may lead to the implementation of enterprise-wide architectures, such as bus architectures, or department-specific architectures, such as independent data marts.
* An urgent need for a data warehouse project, restrictions on economic and human resources, as well as poor IT staff skills may suggest that a type of “quick” architecture, such as independent data marts, should be implemented.
* The minor role played by a data warehouse project in enterprise strategies can make you prefer an architecture type based on independent data marts over a hub-and-spoke architecture type.
* The frequent need for integrating preexisting data warehouses, possibly deployed on heterogeneous platforms, and the pressing demand for uniformly accessing their data can require a federated architecture type.

## Physical Model

The starting point for the physical model is the logical model. The physical model should mirror the logical model as much as possible, although some changes in the structure of the tables and / or columns may be necessary. In addition the physical model will include staging or maintenance tables that are usually not included in the logical model. Although your environment may not have such clearly defined layers you should have some aspects of each layer in your database to ensure it will continue to scale as it increases in size and complexity.

### Staging layer

The staging layer enables the speedy extraction, transformation and loading (ETL) of data from your operational systems into the data warehouse without distributing any of the business users. It is in this layer the much of the complex data transformation and data-quality processing will occur. The tables in the staging layer are normally segregated from the "live" data warehouse. The most basic approach for the staging layer is to have it be an identical schema to the one that exists in the source operational system(s) but with some structural changes to the tables, such as range partitioning. It is also possible that in some implementations this layer is not necessary, as all data transformation processing will be done "on the fly" as data is extracted from the source system before it is inserted directly into the Foundation Layer. Either way you will still have to load data into the warehouse.

#### Efficient Data Loading

Whether you are loading into a staging layer or directly into the foundation layer the goal is to get the data into the warehouse in the most expedient manner. In order to achieve good performance during the load you need to begin by focusing on where the data to-being-loaded resides and how you load it into the database. For example, you should not use a serial database link or a single JDBC connection to move large volumes of data. Flat files are the most common and preferred mechanism of large volumes of data.

#### Staging Area

The area where flat files are stored prior to being loaded into the staging layer of a data warehouse system is commonly known as staging area. The overall speed of your load will be determined by (A) how quickly the raw data can be read from staging area and (B) how fast it can be processed and inserted into the database. It is highly recommended that you stage the raw data across as many physical disks as possible to ensure the reading it is not a bottleneck during the load.

#### Preparing the raw data files

In order to parallelize the data load Oracle needs to be able to logically break up the raw data files into chunks, known as granules. To ensure balanced parallel processing, the number of granules is typically much higher than the number of parallel server processes. At any given point in time, a parallel server process is allocated one granule to work on; once a parallel server process completes working on its granule another granule will be allocated until all of the granules have been processed and the data is loaded.

In order to create multiple granules within a single file, Oracle needs to be able to look inside the raw data file and determine where each row of data begins and ends. This is only possible if each row has been clearly delimited by a known character such as new line or a semicolon.

If a file is not position-able and seek-able, for example the file is compressed (.zip file), then the files cannot be broken up into granules and the whole file is treated as a single granule. Only one parallel server process will be able to work on the entire file.

In order to parallelize the loading of compressed data files you need to use multiple compressed data files. The number of compressed data files used will determine the maximum parallel degree used by the load.

When loading multiple data files (compressed or uncompressed) via a single external table it is recommended that the files are similar in size and that their sizes should be a multiple of 10MB. If different size files have to be used it is recommend that the files are listed from largest to smallest.

By default, Oracle assumes that the flat file has the same character set as the database. If this is not the case you should specify the character set of the flat file in the external table definition to ensure the proper character set conversions can take place.

#### External Tables

Oracle offers several data loading options

* External table or SQL\*Loader
* Oracle Data Pump (import & export)
* Change Data Capture and Trickle feed mechanisms (such as Oracle GoldenGate)
* Oracle Database Gateways to open systems and mainframes
* Generic Connectivity (ODBC and JDBC)

Which approach should you take? Obviously this will depend on the source and format of the data you receive. As mentioned earlier, flat files are the most common mechanism for load large volumes of data. If you are loading from files into Oracle you have two options, SQL\*Loader or external tables. Oracle strongly recommends that you load using external tables rather than SQL\*Loader.

* Unlike SQL\*Loader, external tables allows transparent parallelization inside the database.
* You can avoid staging data and apply transformations directly on the file data using arbitrary SQL or PL/SQL constructs when accessing external tables. SQL Loader requires you to load the data as-is into the database first.
* Parallelizing loads with external tables enables a more efficient space management compared to SQL\*Loader, where each individual parallel loader is an independent database sessions with its own transaction. For highly partitioned tables this could potentially lead to a lot of wasted space.

The most common approach when loading data from an external table is to do a CREATE TABLE AS SELECT (CTAS) statement or an INSERT AS SELECT (IAS) statement into an existing table.

#### Parallel Direct Path Load

The key to good load performance is to use direct path loads wherever possible. A direct path load parses the input data according to the description given in the external table definition, converts data for each input field to its corresponding Oracle data type, then builds a column array structure for the data. These column array structures are used to format Oracle data blocks and build index keys. The newly formatted database blocks are then written directly to the database, bypassing the standard SQL processing engine and the database buffer cache.

A CTAS will always use direct path load but IAS statement will not. In order to achieve direct path load with an IAS statement you must add the APPEND hint to the command.

#### Partition exchange loads

It is strongly recommended that the larger tables or fact tables in a data warehouse are partitioned. One of the benefits of partitioning is the ability to load data quickly and easily with minimal impact on the business users by using the exchange partition command. The exchange partition command allows you to swap the data in a non-partitioned table into a particular partition in your partitioned table. The command does not physically move data; instead it updates the data dictionary to exchange a pointer from the partition to the table and vice versa. Because there is no physical movement of data, an exchange does not generate redo and undo, making it a sub-second operation and far less likely to impact performance than any traditional data-movement approaches such as INSERT.

#### Data Compression

Another key decision that you need to make during the load phase is whether or not to compress your data. Using table compression obviously reduces disk and memory usage, often resulting in better scale-up performance for read-only operations. Table compression can also speed up query execution by minimizing the number of round trips required to retrieve data from the disks. Compressing data however imposes a performance penalty on the load speed of the data. The overall performance gain typically outweighs the cost of compression.

If you decide to use compression, consider sorting your data before loading it to achieve the best possible compression rate. The easiest way to sort incoming data is to load it using an ORDER BY clause on either your CTAS or IAS statement. You should ORDER BY a NOT NULL column (ideally non numeric) that has a large number of distinct values (1,000 to 10,000).

### Foundation layer - Third Normal Form

From staging, the data will transition into the foundation or integration layer via another set of ETL processes. Data begins to take shape and it is not uncommon to have some end-user application access data from this layer especially if they are time sensitive, as data will become available here before it is transformed into the dimension / performance layer. Traditionally this layer is implemented in the Third Normal Form (3NF).

There are several approaches to optimize 3NF schema. The larger tables should be partitioned using composite partitioning (range-hash or list-hash). There are three reasons for this:

1. Easier manageability of terabytes of data
2. Faster accessibility to the necessary data
3. Efficient and performant table joins

Finally Parallel Execution enables a database task to be parallelized or divided into smaller units of work, thus allowing multiple processes to work concurrently. By using parallelism, a terabyte of data can be scanned and processed in minutes or less, not hours or days.

### Access layer - Star Schema

The access layer represents data, which is in a form that most users and applications can understand. It is in this layer you are most likely to see a star schema.

Tuning a star query is very straight forward. The two most important criteria are:

* Create a bitmap index on each of the foreign key columns in the fact table or tables
* Set the initialization parameter STAR\_TRANSFORMATION\_ENABLED to TRUE. This will enable the optimizer feature for star queries which is off by default for backward compatibility.

## Meta-data

The term meta-data can be applied to the data used to define other data. In the scope of data warehousing, meta-data plays an essential role because it specifies source, values, usage, and features of data warehouse data and defines how data can be changed and processed at every architecture layer. Figures 1 and 2 show that the meta-data repository is closely connected to the data warehouse. Actually, according to Kimball, metadata is everything, except for the data itself. Applications use it intensively to carry out data-staging and analysis tasks.

You can classify meta-data into two partially overlapping categories. This classification is based on the ways system administrators and end users exploit meta-data. System administrators are interested in internal meta-data (back room metadata) because it defines data sources, transformation processes, population policies, logical and physical schemata, constraints, and user profiles. External meta-(front room metadata) data is relevant to end users. For example, it is about definitions, quality standards, units of measure, relevant aggregations.

# Accessing Data Warehouses

Analysis is the last level common to all data warehouse architecture types. After cleansing, integrating, and transforming data, you should determine how to get the best out of it in terms of information. The following sections show the best approaches for end users to query data warehouses: reports, OLAP, and dashboards. End users often use the information stored to a data warehouse as a starting point for additional business intelligence applications, such as what-if analyses and data mining.

## Reports

This approach is oriented to those users who need to have regular access to the information in an almost static way. For example, suppose a local health authority must send to its state offices monthly reports summing up information on patient admission costs. The layout of those reports has been predetermined and may vary only if changes are applied to current laws and regulations. Designers issue the queries to create reports with the desired layout and “freeze” all those in an application. In this way, end users can query current data whenever they need to.

A report is defined by a query and a layout. A query generally implies a restriction and an aggregation of multidimensional data. For example, you can look for the monthly receipts during the last quarter for every product category. A layout can look like a table or a chart (diagrams, histograms, pies, and so on).

A reporting tool should be evaluated not only on the basis of comprehensive report layouts, but also on the basis of flexible report delivery systems. A report can be explicitly run by users or automatically and regularly sent to registered end users. For example, it can be sent via e-mail.

Keep in mind that reports existed long before data warehouse systems came to be. Reports have always been the main tool used by managers for evaluating and planning tasks since the invention of databases. However, adding data warehouses to the mix is beneficial to reports for two main reasons: First, they take advantage of reliable and correct results because the data summed up in reports is consistent and integrated. In addition, data warehouses expedite the reporting process because the architectural separation between transaction processing and analyses significantly improves performance.

## OLAP

OLAP might be the main way to exploit information in a data warehouse. Surely it is the most popular one, and it gives end users, whose analysis needs are not easy to define beforehand, the opportunity to analyze and explore data interactively on the basis of the multidimensional model. While users of reporting tools essentially play a passive role, OLAP users are able to start a complex analysis session actively, where each step is the result of the outcome of preceding steps. Real-time properties of OLAP sessions, required in-depth knowledge of data, complex queries that can be issued, and design for users not familiar with IT make the tools in use play a crucial role. The GUI of these tools must be flexible, easy-to-use, and effective.

An OLAP session consists of a navigation path that corresponds to an analysis process for facts according to different viewpoints and at different detail levels. This path is turned into a sequence of queries, which are often not issued directly, but differentially expressed with reference to the previous query. The results of queries are multidimensional. Because we humans have a difficult time deciphering diagrams of more than three dimensions, OLAP tools typically use tables to display data, with multiple headers, colors, and other features to highlight data dimensions.

## Dashboards

Dashboards are another method used for displaying information stored to a data warehouse. The term dashboard refers to a GUI that displays a limited amount of relevant data in a brief and easy-to-read format. Dashboards can provide a real-time overview of the trends for a specific phenomenon or for many phenomena that are strictly connected with each other. The term is a visual metaphor: the group of indicators in the GUI are displayed like a car dashboard. Dashboards are often used by senior managers who need a quick way to view information. However, to conduct and display very complex analyses of phenomena, dashboards must be matched with analysis tools. Today, most software vendors offer dashboards for report creation and display.

Keep in mind, however, that dashboards are nothing but performance indicators behind GUIs. Their effectiveness is due to a careful selection of the relevant measures, while using data warehouse information quality standards. For this reason, dashboards should be viewed as a sophisticated effective add-on to data warehouse systems, but not as the primary goal of data warehouse systems. In fact, the primary goal of data warehouse systems should always be to properly define a process to transform data into information.

# OLAP, MOLAP, and HOLAP

These three acronyms conceal three major approaches to implementing data warehouses, and they are related to the logical model used to represent data:

* ROLAP stands for Relational OLAP, an implementation based on relational DBMSs.
* MOLAP stands for Multidimensional OLAP, an implementation based on multidimensional DBMSs.
* HOLAP stands for Hybrid OLAP, an implementation using both relational and multidimensional techniques.

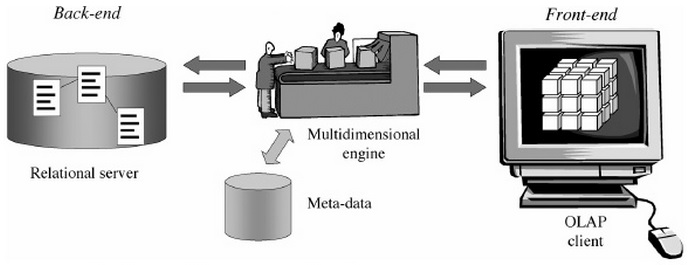
The idea of adopting the relational technology to store data to a data warehouse has a solid foundation if you consider the huge amount of literature written about the relational model, the broadly available corporate experience with relational database usage and management, and the top performance and flexibility standards of relational DBMSs (RDBMSs). The expressive power of the relational model, however, does not include the concepts of dimension, measure, and hierarchy, so you must create specific types of schemata so that you can represent the multidimensional model in terms of basic relational elements such as attributes, relations, and integrity constraints. This task is mainly performed by the well-known star schema.

The main problem with ROLAP implementations results from the performance hit caused by costly join operations between large tables. To reduce the number of joins, one of the key concepts of ROLAP is denormalization—a conscious breach in the third normal form oriented to performance maximization. To minimize execution costs, the other key word is redundancy, which is the result of the materialization of some derived tables (views) that store aggregate data used for typical OLAP queries.

Different from a ROLAP system, a MOLAP system is based on an ad hoc logical model that can be used to represent multidimensional data and operations directly. The underlying multidimensional database physically stores data as arrays and the access to it is positional. Grid-files (Nievergelt et al., 1984; Whang and Krishnamurthy, 1991), R\*-trees (Beckmann et al., 1990) and UB-trees (Markl et al., 2001) are among the techniques used for this purpose.

The greatest advantage of MOLAP systems in comparison with ROLAP is that multidimensional operations can be performed in an easy, natural way with MOLAP without any need for complex join operations. For this reason, MOLAP system performance is excellent.

The intermediate architecture type, HOLAP, aims at mixing the advantages of both basic solutions. It takes advantage of the standardization level and the ability to manage large amounts of data from ROLAP implementations, and the query speed typical of MOLAP systems. HOLAP implies that the largest amount of data should be stored in an RDBMS to avoid the problems caused by sparsity, and that a multidimensional system stores only the information users most frequently need to access. If that information is not enough to solve queries, the system will transparently access the part of the data managed by the relational system.



**Figure 5 ROLAP Architecture**

# Additional Issues

The issues that follow can play a fundamental role in tuning up a data warehouse system. These points involve very wide-ranging problems and are mentioned here to give you the most comprehensive picture possible.

## Quality

In general, we can say that the quality of a process stands for the way a process meets users’ goals. In data warehouse systems, quality is not only useful for the level of data, but above all for the whole integrated system, because of the goals and usage of data warehouses. A strict quality standard must be ensured from the first phases of the data warehouse project.

Defining, measuring, and maximizing the quality of a data warehouse system can be very complex problems. For this reason, we mention only a few properties characterizing data quality here:

* Accuracy. Stored values should be compliant with real-world ones.
* Freshness. Data should not be old.
* Completeness. There should be no lack of information.
* Consistency. Data representation should be uniform.
* Availability. Users should have easy access to data.
* Traceability. Data can easily be traced data back to its sources.
* Clearness. Data can be easily understood.

Technically, checking for data quality requires appropriate sets of metrics. In the following sections, we provide an example of the metrics for a few of the quality properties mentioned:

* Accuracy and completeness. Refers to the percentage of tuples not loaded by an ETL process and categorized on the basis of the types of problem arising. This property shows the percentage of missing, invalid, and nonstandard values of every attribute.
* Freshness. Defines the time elapsed between the date when an event takes place and the date when users can access it.
* Consistency. Defines the percentage of tuples that meet business rules that can be set for measures of an individual cube or many cubes and the percentage of tuples meeting structural constraints imposed by the data model (for example, uniqueness of primary keys, referential integrity, and cardinality constraint compliance).

Note that corporate organization plays a fundamental role in reaching data quality goals. This role can be effectively played only by creating an appropriate and accurate certification system that defines a limited group of users in charge of data. For this reason, designers must raise senior managers’ awareness of this topic. Designers must also motivate management to create an accurate certification procedure specifically differentiated for every enterprise area. A board of corporate managers promoting data quality may trigger a virtuous cycle that is more powerful and less costly than any data cleansing solution. For example, you can achieve awesome results if you connect a corporate department budget to a specific data quality threshold to be reached.

An additional topic connected to the quality of a data warehouse project is related to documentation. Today most documentation is still nonstandardized. It is often issued at the end of the entire data warehouse project. Designers and implementers consider documentation a waste of time, and data warehouse project customers consider it an extra cost item. Software engineering teaches that a standard system for documents should be issued, managed, and validated in compliance with project deadlines. This system can ensure that different data warehouse project phases are correctly carried out and that all analysis and implementation points are properly examined and understood. In the medium and long term, correct documents increase the chances of reusing data warehouse projects and ensure project know-how maintenance.

## Security

Information security is generally a fundamental requirement for a system, and it should be carefully considered in software engineering at every project development stage from requirement analysis through implementation to maintenance. Security is particularly relevant to data warehouse projects, because data warehouses are used to manage information crucial for strategic decision-making processes. Furthermore, multidimensional properties and aggregation cause additional security problems similar to those that generally arise in statistic databases, because they implicitly offer the opportunity to infer information from data. Finally, the huge amount of information exchange that takes place in data warehouses in the data-staging phase causes specific problems related to network security.

Appropriate management and auditing control systems are important for data warehouses. Management control systems can be implemented in front-end tools or can exploit operating system services. As far as auditing is concerned, the techniques provided by DBMS servers are not generally appropriate for this scope. For this reason, you must take advantage of the systems implemented by OLAP engines. From the viewpoint of users’ profile–based data access, basic requirements are related to hiding whole cubes, specific cube slices, and specific cube measures. Sometimes you also have to hide cube data beyond a given detail level.

## Evolution

Many mature data warehouse implementations are currently running in midsize and large companies. The unstoppable evolution of application domains highlights dynamic features of data warehouses connected to the way information changes at two different levels as time goes by:

* Data level. Even if measured data is naturally logged in data warehouses thanks to temporal dimensions marking events, the multidimensional model implicitly assumes that hierarchies are completely static. It is clear that this assumption is not very realistic. For example, a company can add new product categories to its catalog and remove others, or it can change the category to which an existing product belongs in order to meet new marketing strategies.
* Schema level. A data warehouse schema can vary to meet new business domain standards, new users’ requirements, or changes in data sources. New attributes and measures can become necessary. For example, you can add a subcategory to a product hierarchy to make analyses richer in detail. You should also consider that the set of fact dimensions can vary as time goes by.

Temporal problems are even more challenging in data warehouses than in operational databases, because queries often cover longer periods. For this reason, data warehouse queries frequently deal with different data and/or schema versions. Moreover, this point is particularly critical for data warehouses that run for a long time, because every evolution not completely controlled causes a growing gap between the real world and its database representation, eventually making the data warehouses obsolete and useless.

# The Interim Results

To compare effectiveness and success of different architectures implementations a web-based survey—targeted at individuals involved in an organization’s data warehouse implementation used to collect data. The survey included questions about the respondent, the respondent’s company, the company’s data warehouse, and the success of the data warehouse architecture. Four hundred fifty-four respondents provided usable information.

Surveyed companies ranged from small (less than $10 million in revenue) to large (in excess of $10 billion). Most of the companies are located in the United States (60%) and represent a variety of industries, with the financial services industry (15%) providing the most responses.

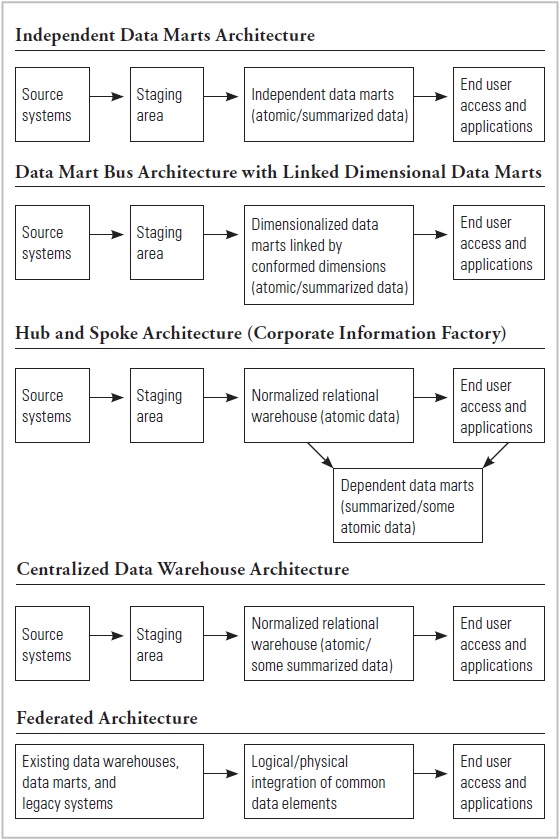
The predominant architecture was the hub-and-spoke (39%), followed by the bus architecture (26%), centralized (17%), independent data marts (12%), and federated (4%). The most common platform for hosting the data warehouses was Oracle (41%), followed by Microsoft (19%) and IBM (18%). The average (mean) gross revenue varied from $3.7 billion for independent data marts to $6 billion for the federated architecture.

Four measures were used to assess the success of the architectures: (1) information quality, (2) system quality, (3) individual impacts, and (4) organizational impacts. The questions used a seven-point scale, with the higher score indicating a more successful architecture. Figure 7 shows the average scores for the measures across the architectures

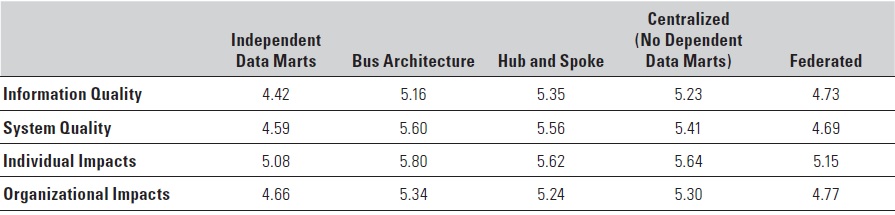
Independent data marts scored the lowest on all measures. This finding confirms the conventional wisdom that independent data marts are a poor architectural solution.

Next lowest on all measures was the federated architecture. Firms sometimes have disparate decision-support platforms resulting from mergers and acquisitions, and they may choose a federated approach, at least in the short run. The findings suggest that the federated architecture is not an optimal long-term solution.

What is interesting, however, is the similarity of the averages for the bus, hub-and-spoke, and centralized architectures. The differences are sufficiently small that no claims can be made for a particular architecture’s superiority over the others, at least based on a simple comparison of these success measures.



**Figure 6 The five data warehouse architectures studied.**



**Figure 7 The success of the five architectures.**

The similar success of the bus, hub-and-spoke, and centralized architectures is perhaps not all that surprising. In some ways, the architectures have evolved over time and become more similar. For example, the hub-and-spoke architecture often includes dimensional data marts, which is at the heart of the bus architecture. Even the development methodologies (e.g., top down for the hub-and-spoke and centralized architectures, and life cycle or bottom up for the bus architecture) have evolved and become more similar. Each stresses the need to start small and deliver short-term “wins” but have a long-term plan. This evolution is appropriate and good for the industry, but it is also a likely reason that the scores on the success metrics are similar.

# Source Books and Articles

1. Golfarelli M., & Rizzi S. Data Warehouse Design: Modern Principles and Methodologies. New York City: McGraw-Hill Osborne Media, 2009.
2. Colgan M. Best Practices for a Data Warehouse on Oracle Database 11g; An Oracle White Paper. Redwood City: Oracle, 2010.
3. Watson H.J., & Ariyachandra T. Data Warehouse Architectures: Factors in the Selection Decision and the Success of the Architectures, 2005.
4. Powell G. Oracle Data Warehouse Tuning for 10g. Oxford: Elsevier Digital Press, 2005.
5. Lane, P. Oracle Database Data Warehousing Guide, 11g Release 2 (11.2). Redwood City: Oracle, 2013.
6. Kimball R., & Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Third Edition. Indianapolis: John Wiley & Sons, 2013.