Data Warehouses as Databases of Special Purpose. Business Intelligence

Data warehouses are databases of a very special purpose, and this defines their properties and approaches to their design. As with any database, when designing a data warehouse we try to satisfy user requirements on completeness and consistency of data, performance, and convenience of use. We also want a database to be easy to implement and support.

We want to show that the methodology of design of these special databases grew out from the traditional methodology of database design.

1.1. Evolution of a Data Warehouse

Increasing capabilities of computers and advancements in storage capacity led to wide use of databases for support of different business activities. Eventually, businesses accumulated significant amount of business data. It was data accumulated historically for each application, as well as for different applications across the company (Figure 1-1).

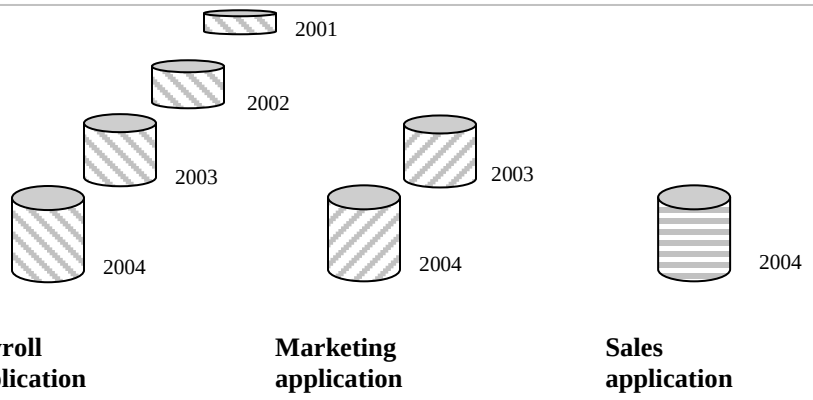


Figure 1-1. Accumulating historical data across business applications.

Businesses tried to take advantage of accumulated data. In addition to regular reports on current business performance (e.g. daily, monthly, quarterly, or yearly reports) executives started requesting from the company's databases various retrospective reports to analyze business performance over the years. Such analysis helped to understand the business tendencies and factors that influence the business, and to improve quality of business decisions. For example, if reports about sales of products of different categories on `Sales` database showed that for the last several years sales of swimwear dropped significantly during winter, then this could prompt managers to stock less swimwear for future winters months. In many cases, historical data was kept in separate archive

databases, because it was not immediately used by applications that ran the business and keeping it in applications' databases created performance and storage problems.

Usually, such retrospective analytical requests resulted in processing of huge amounts of data and aggregating it in different ways. If computing of the commonly used aggregates required significant resources and time, then the aggregates were pre-calculated and stored separately. For example, if managers were trying to see tendencies of sales of different categories of products (e.g. swimwear) during different seasons without getting into details of sales of particular products, then it was reasonable to have data about sales of categories of products prepared for such analysis. Such aggregate data was obtained from detailed data about sales of every product supported by the `Sales` application.

More comprehensive and global analysis of the company's business required involving data from different applications (e.g. `Marketing`); that is why additional data from various applications was exported to the database with historical and aggregated data.

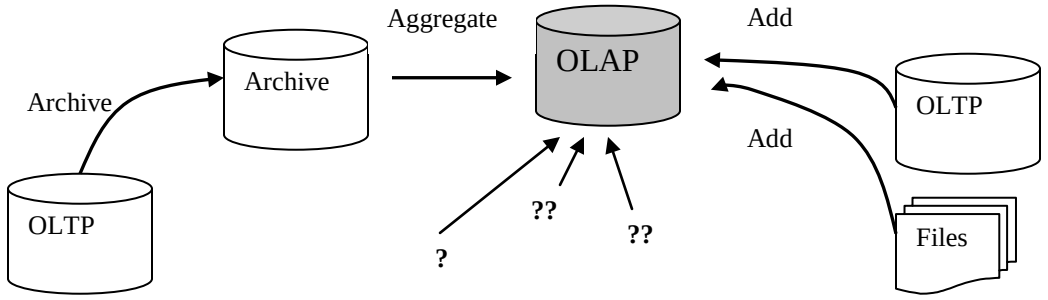


Figure 1-2. Preparing data for specific analytical needs.

That is how a separate database with historical and aggregated data from possibly different databases or other data sources appeared (see Figure 1-2). The purpose of this database defines its name--*On-Line Analytic Processing (OLAP)* database or data warehouse, as opposed to *On-Line Transaction Processing (OLTP)* database, which is used for running the business.

1.2. Basic Features of OLAP databases

There are several important differences between OLAP and OLTP databases. These differences originate from different types of users with specific requirements. An OLTP database is used by data entry people who maintain data in the database and low-level managers who routinely retrieve this data. An OLAP database serves high-level managers responsible for making strategic business decisions who, by analyzing historical business data, expect to understand better inner mechanisms of the business and improve quality of the decisions.

The described purpose of an OLAP database is translated into the following very specific requirements for a database designer:

- *Performance of reading requests to an OLAP database is of the main importance.* Users need only to retrieve data (remember, data is not directly entered in data warehouses — it is the pre-processed in some way data from various primary data sources of the company). This is different from OLTP databases, which have to provide satisfactory performance for both, effective data entry and data retrieval.
- *Most of requests to an OLAP database have "ad hoc" character*. When looking for data to confirm or reject their decision users can have different and unexpected requests to a data warehouse. Maintaining ad hoc requests on a database has several problems. First, users may require assistance of a database programmer in building such requests, and second, performance on a data warehouse can be unsatisfactory because it is impossible to tune the database for each and all of these requests. In OLTP databases, on the other hand, users are either entering data with the help of predefined procedures, or are retrieving data by predefined patterns. This allows database programmers to define routines of data access and create tools (forms for data entry and reports) through which users access a database, as well as tune the database for these specific routines.

Traditional methodology of database design was initially developed for OLTP databases and it does not serve the goals of design of OLAP databases. Methodology of OLTP database design is aimed on building normalized databases which usually contain numerous interrelated tables (see Appendix D with the short review of the methodology of database design). Normality of a database guarantees from main anomalies of data modifications and improves consistency of data. Retrieving data from such database usually involves several tables, and each of multi-table queries requires special measures (database tuning) to make its performance acceptable to the users. Neither a database nor queries on this database have some regular structure, and users need special applications that reflect the peculiarities of each database and implement patterns of access to data.

Because traditionally designed databases were not effective for analytical use, they had to undergo various modifications. The main ideas behind the modifications were to improve performance of all (or most of) queries on a database, and to make the structure of this database more regular to simplify constructing ad hoc queries and eliminate the dependency of users of the OLAP database on programmers. Systematization of successful approaches to produce databases with these qualities led to a special methodology of design of OLAP databases.

1.3. Moving from OLTP to OLAP

To understand the basic points of the methodology of data warehouse design we will discuss an OLTP database for a chain of retail stores and decide how to modify the design of the database to make it better satisfy analytical needs.

The retail company needs to keep in the database records about sales of products in different stores. For each product, we need to know its name and description; as an identifier of a product, we choose Stock Keeping Unit (SKU). Products are classified into subcategories; each subcategory has a name. Each product has one subcategory, and there can be multiple products of the same subcategory. Subcategories are organized into categories, and each category has a name. Subcategory can belong to one category, and

one category can include multiple subcategories. For stores we need to keep store number and name. Stores are included in sales regions, where each region is defined by region code and name. Each store belongs to one region, and one region can include several stores. Sales regions, in turn, are included in sales districts, about which we know their code and name. A district can contain several regions, while a region is assigned to one district only.

The database also contains data about other activities of the company, e.g. ordering and purchasing of products, which we omit for simplicity.

After we add attributes to subcategory and category to serve as identifiers, the conceptual model of our database will be presented with Entity- Relational Diagram (ERD) in Figure 1-3:

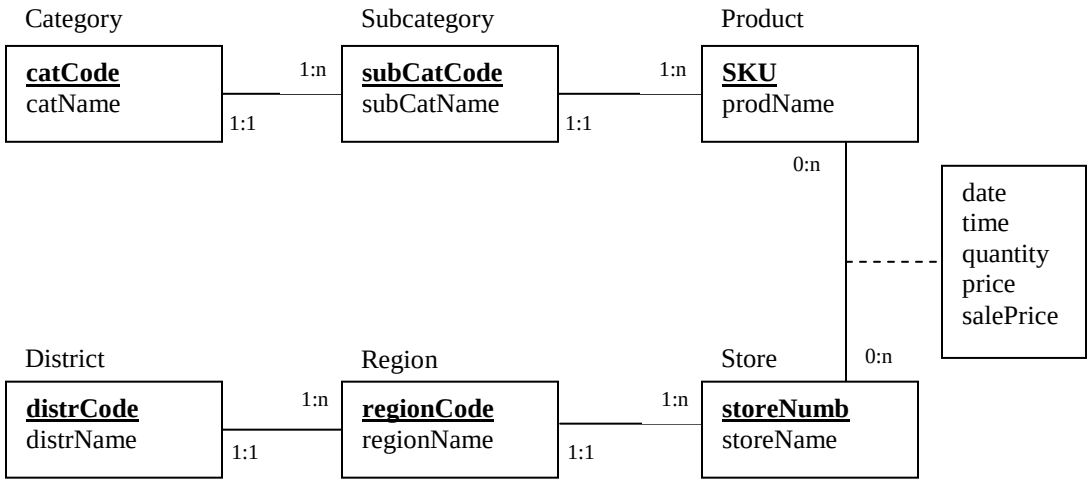


Figure 1-3. ERD of OLTP database.

Mapping the diagram into the relational model of the database gives the six relations for the entities and one relation for the many-to-many relationship between `Store` and `Product`, which we will name `Sales` (primary keys of relations are underlined, and foreign keys are in italic):

```
Category (catCode, catName)
Subcategory (subCatCode, subCatName, catCode)
Product (SKU, prodName, subCatCode)
District (distrCode, distrName)
Region (regionCode, regionName, distrCode)
Store (storeNumb, storeName, regionCode)
Sales (storeNumb, SKU, date, time, quantity, price,
salePrice)
```

As we mentioned, to better understand business mechanisms and to make sound business decisions users want to analyze data about performance of the business. They know how to measure performance of the business, and their main question is "Why?"--why performance was better or worse, e.g. "Why sales of product A dropped in the first quarter of the current year comparing to sales of this product last year?", or "Why during the last three years sales region X was less profitable than other regions?" (sales and profitability are used here as measures of the business).

Data itself cannot give a direct answer to this question. But the database can answer questions like "Who?", "What?", "Where?", "When?", and possibly some other, which allow users to either answer this main question, or to get some ideas about the possible answer; the ideas could be supported or rejected by additional questioning of data.

Here is a possible scenario of a user working with the database while trying to understand problems of the business:

1. The user finds out that sales of a particular product were unusually low and compares them to sales of this product during the last year:
   - What were sales of product A in the first quarter of the current year?
   - What were sales of product A in the first quarter of the last year?
2. This year sales are really significantly lower than before, and the user tries to look for possible explanations, e.g. different pricing policies:
   - What was the average price of product A in the first quarter of the current year?
   - What was the average price of product A in the first quarter of the last year?
3. The pricing policy of the current year was similar to the pricing policy of the last year, and there is still no answer to the main question. The user continues trying to understand the reasons and checks sales of other products of the same subcategory:
   - How sales of product B, which falls in the same subcategory as product A, changed compared to its sales last year?
4. The sales of product B also dropped, and the user suggests that maybe sales of all products of the subcategory dropped, which is confirmed by further requests to the database for all products sold in both years.
5. The user checks sales of a new product of the same subcategory (which appeared in the stores only in the current year) and finds out that sales of this product were very high and that total sales of products of the subcategory were comparable with corresponding sales last year.

The database access scenario here is absolutely different from the scenario on an OLTP database – the user doesn't know what the next step would be, and the next step depends on results of the previous steps or is just a guess. While users of an OLTP database ask similar questions, the routine of asking them is perfectly well defined and therefore supported by convenient tools and appropriate tuning of the database. For example, once a quarter users of `Sales` OLTP database request a report on quarterly sales of products,

or subcategories of products. Also, OLTP requests usually refer to the latest business data.

Because of the supporting database applications, users of the OLTP database do not need to know about the actual database structure, they see the database the way it is presented in forms and reports, and follow routines of data entry and retrieval; we say that the database is transparent to them. Ad hoc nature of requests to the OLAP database does not allow users to rely on the tools and requires from them knowledge of actual database structure or continuous assistance of database professionals. For example, if the user asks about sales of a *particular product* in some *stores* during a *period of time*, he should know that tables `Product`, `Store`, and `Sales` must be involved in the processing of his request and how to join them. For the request about sales of *categories* of products in stores of some particular *district* the user has to use all seven tables of our database with the appropriate join conditions.

Though the structure of these two mentioned requests is different (they use different sets of tables), they are very similar in nature. We may say that in each case users want to analyze business performance across several dimensions: one of them defines *what* was sold (particular products, or products of specific categories, or products of given subcategories), another-- *where* it was sold (stores, or stores of a particular region or district), and these two dimensions are defined by products and stores.

It is natural for users to see subcategory and category as features of product without knowing about relationships between subcategories, categories, and products. We can move to a single table with all data about products required for analysis if we merge table `Category` with table `Subcategory`, and then merge the resulting table with table `Product`,:

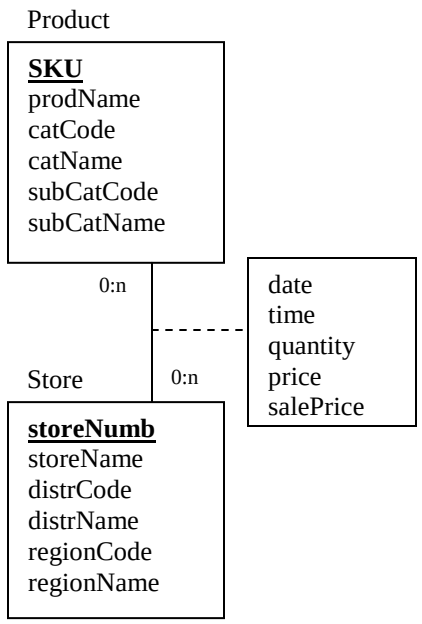
```
Product (SKU, prodName, subCatCode, subCatName,
catCode, catName)
```

We, actually, performed denormalization of the database – an approach we often use to simplify the structure of an OLTP database and improve performance of some queries. You can imagine that OLAP database will be of a considerable size (history of the business for several years), and that performance of reading queries involving multiple tables could be a problem. Denormalization leads to the database with fewer tables and better performing read queries. Therefore, the result of our attempt to simplify the structure of data also promises a better performance.

Similarly, we may combine tables `District`, `Region`, and `Store`:

```
Store (storeNumb, storeName, regionCode, regionName,
distrCode, distrName)
```

Our database becomes very simple and obvious to users (see Figure 1-4).




```
Product (SKU, prodName, subCatCode,
      subCatName, catCode, catName)

Store (storeNumb, storeName,
      regionCode,        regionName,
      distrCode, distrName)

Sales (storeNumb, SKU, date, time,
      quantity, price, salePrice)
```

Figure 1-4. Simplicity of the denormalized database.

On this simplified database any request about sales will result in a query on not more than three tables (instead of seven) with the same conditions of joining the tables – the queries now have a regular structure. Example of a query on the denormalized database is shown in Figure 1-5 (the regular or core part of queries is in bold).

```
SELECT SUM(quantity)

FROM Sales s, Product p, Store st

WHERE s.SKU = p.SKU AND
      s.storeNumb = st.storeNumb

          AND p.catName = 'X'
          AND st.distrName= 'Y';
```

Variable result

Permanent "core" of queries

Variable conditions of analyses

Figure 1-5. Regular structure of queries

Simple structure of the database and queries results in a better performance of read queries, and it allows developing and successfully using special applications that help users to specify their ad hoc requests to the database.

What about normalization, the goal of design of OLTP databases, which we gave up so easily? Normalization protects a database from modification anomalies and it looses its importance in data warehouses, which are read-only databases. By giving up not too important normalization we gain simplicity and better performance.

1.4. Basic Points of Design Methodology

We were discussing design of a database, which keeps historical data about business performance and allows to easily analyze this data in many different ways. In our case, data about performance resides in table `Sales`, and for measures of such performance users chose `quantity` and `price`. In data warehouses, table that contains data about business performance is called *`Fact`* table; measures of performance included in `Fact` table are called *facts*.

Our Fact table is joined to two tables that provide additional descriptions of business performance: *what* was sold (product, subcategory, and category) and *where* the sale took place (store, region, and district). These two tables are called *dimensions*. By applying different constraints to dimensional attributes users can get different views of business performance, e.g. `Category = 'X'`, `District = 'Y'`, or `Category = 'X'AND District = 'Y'`.

The methodology of data warehouse design uses a special graphical tool for data modeling. It is called the *dimensional* or *star schema* or model. The dimensional schema of the database for our case is shown in Figure 1-6:

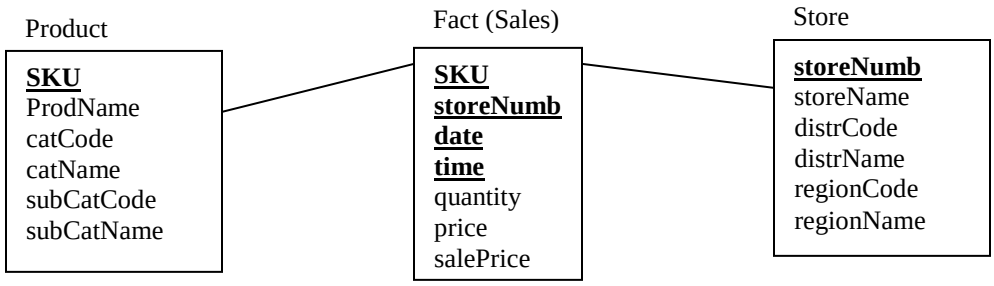


Figure 1-6. Dimensional model of Sales case.

Basic features of dimensional model are:

- *Simple regular architecture*, which gave name "star schema" to the model (see Figure 1-7). Note that schemas in Figures 1-6 and 1-7 present not conceptual, but logical models of data. Simple and regular structure of OLAP database allows to start the design with the logical model.

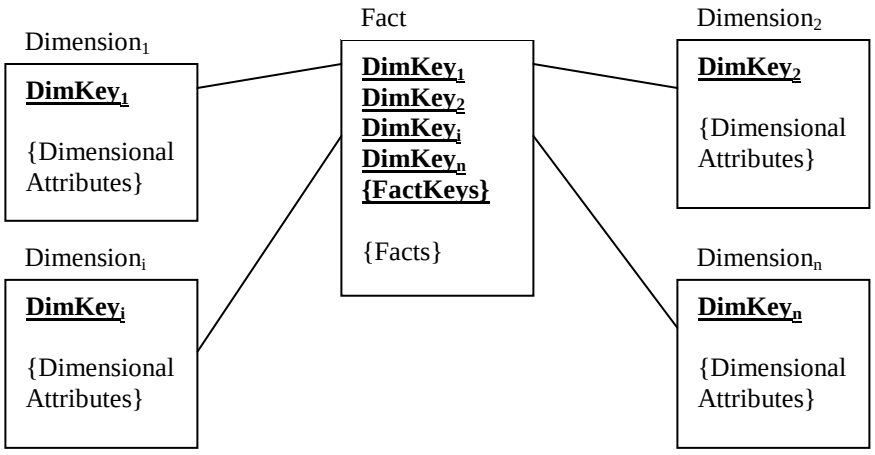


Figure 1-7. The general dimensional model.

- *Fact table*. The central table of star schema that contains data (facts) about business performance is called `Fact` table.
- *Dimensions*. Fact table is linked to the tables that describe business performance to the extent needed for analyses; these tables are called dimensions (another name of the model is "dimensional"). In our case, we have dimension `Product` with attributes `catName` and `subCatName` for category and subcategory respectively because users need to analyze sales of categories and subcategories of products as well as sales of separate products. Users did not specify that they would be interested in analyzing the business in other contexts, e.g. for diet and not diet products, but if they did, then we would have to provide possibility of such analysis by including dietType attribute in `Product` dimension.

Anything users need to know about the business to be able to analyze it is either facts or dimensional features of the facts. It means that architecture of a data warehouse does not become more complicated – it is always the `Fact` table linked to `dimensional` tables, and dimensional tables linked only to `Fact`. This guarantees simplicity of the model, and as a result, simplicity and regular structure of queries to the database. For example, requesting different totals across several dimensions will be expressed with the help of queries of the following structure:

```
SELECT <aggregate functions on facts>
FROM Fact f, Dimension_i d_i, …, Dimension_j d_j
WHERE f.d_i_key = d_i.key AND
      . . .
      f.d_j_key = d_j.key AND
      d_i = <constant_i> AND
      . . . AND
      d_j = <constant_j>;
```

In this query, various aggregates are calculated across dimensions $Dimension_i$, …, $Dimension_j$, which are constrained. For example, if we need total quantity of diary products sold in stores of Northeast, we will use aggregate function SUM(quantity)and constrain dimensions Product and Store in the following way: Product.category = 'Diary' and Store.distrName = 'Northeast'.

Part of conditions of the above query shown in bold specifies conditions of join of Fact table with the dimensions used in the query similarly to join of Sales, Product, and Store tables in the query of Figure 1-5. Further in the text, for queries on the dimensional model we will simply use expression <join conditions> to show join conditions on Fact table and the dimensions:

```
SELECT <aggregate functions on facts>
FROM Fact f, Dimension_i d_i, …, Dimension_j d_j
     WHERE <join conditions> AND
     d_i = <constant_i> AND
     . . . AND
     d_j = <constant_j>;
```

The methodology of data warehouse design discourages from so-called "snowflaking", when a dimension participates in relationships with tables other than Fact table (see Figure 1-8).

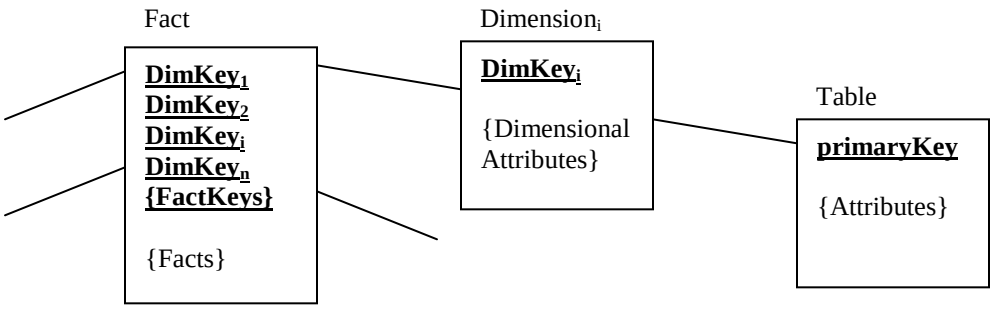


Figure 1-8. Snowflake dimension

1.5. Data Warehouse as a Part of Business Intelligence

Several years ago data warehouses were getting a lot of attention from researchers, vendors, and companies. But failures of many data warehouse projects shifted the focus to what is called today "Business Intelligence" (BI) of a company. It appeared that the main difficulty was not to design a data warehouse, but to sustain it and to be able to utilize its data effectively. Environment of a data warehouse, which keeps it "alive" and makes the business more "intelligent," is shown in Figure 1-9. It includes back-end tools for support of extraction, transformation, and load (ETL) of data from primary data

sources that feed data warehouses (often called the staging area of a data warehouse), and special applications for utilization of data on the front-end. The front-end tools include data mining applications (DMA), decision support systems (DSS), and some other applications. Integrating all these tools with a data warehouse is the most challenging problem in establishing the business intelligence infrastructure of a company.

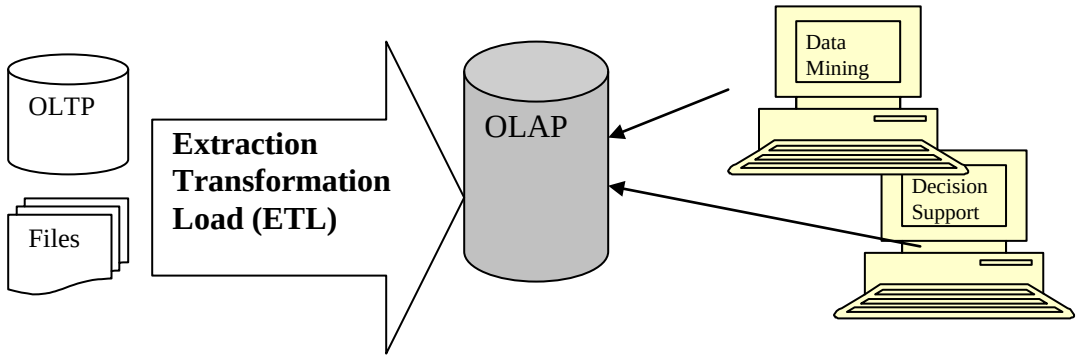


Figure 1-9. Architecture of business intelligence of a company.

1.6. Summary

This handout explains differences between OLTP databases that support everyday business and OLAP databases that enforce strategic decision making. The table below summarizes these differences:

Category	OLTP	OLAP
Type of users' access to data	Read-Write	Read-only
Nature of users' requests	Predefined	Ad hoc
Typical user	Data entry clerk or manager of low level	Manager of high level
Typical application	Data entry application or a standard report	Data Mining System, Decision Support System, and Query Builder
Scope of request	Recent business activities	Business activities for up to several previous years
Level of details	Details of every business transaction or specific aggregates of business data	Various aggregated measurements of business performance

Nature and purpose of OLAP databases are better served by a special model of data, which is called "star" or "dimensional". Dimensional model includes the `Fact` table with measurements of business performance, which is related to dimensional tables that describe these measurements. For example, `Fact` table of `Sales` case discussed in this handout records quantities of sold products. For each sale, if needed, from `Product`

dimension users can get information about various features of sold product, such as subcategory, category, diet type, and others.

Advantages of star or dimensional model are simple data structure and correspondingly simple and regular structure of most of the possible queries. Dimensional model is non-normalized and results in fewer tables than traditional normalized model of data, and this promises improvement of performance. Because users of OLAP database do not modify (insert, update, and delete) but only read data, consistency of a non-normalized OLAP database is not compromised.

More detailed discussion of methodology of dimensional design includes such issues as:

- *The scope of a separate data warehouse.* We mentioned that our OLTP database contained some other interesting data, for example, about ordering and purchasing of products in stores, or promoting products in stores' neighborhoods. What data from data sources should be transferred to a single data warehouse?
- *The optimal design of a data warehouse.* In our case we went from seven tables to three, which simplified the structure of our database and promised a gain in performance of queries. Why did we stop there? Why did not we continue merging tables `Sales`, `Product` and `Store`?
- *Level of details about the business.* In the case discussed in this handout we preserved the level of details as it was in the OLTP database—we recorded each sale transaction. Such approach gave us the most detailed view of the business. But is this unconditionally good and do we always need this?
- *Distribution of data between `Fact` and dimensions.* We said that `Fact` table records facts about business performance and dimensions contain additional descriptions of performance. What data belongs to measures, and what--to descriptions of the business?
- *Measures of business performance.* What are "good" facts? In our case we chose as facts quantity and price of sold products. Are those the best facts to measure and analyze the business?
- *Way of recording business performance.* In our case we recorded each sale transaction. Maybe, we could benefit from recording inventory levels of products in stores at definite moments of time, or from accumulating quantities of sold products in stores?
- *Retrospectiveness of a data warehouse.* In this handout, the only changes to business we wanted to keep track of were sales of products. But what if dimensional data also changed, e.g. the name of a particular product? Should we try to record this change, and if we should, how to do this?
- *How to minimize the size of the data warehouse without sacrificing its functionality.* Data warehouses are very large databases; appropriate design of the tables can help in resolving the size issue.

Review Questions

1. What are main differences between OLTP and OLAP databases?
2. How dimensional model serves analytical needs of users?

3. Why model of OLAP database is called “star” or “dimensional”?
4. What is the role of `Fact` table?
5. What is “snowflaking” and why it is not recommended in the schema of an OLAP database?
6. What is the role of dimensions?
7. Why normalization loses its importance in OLAP database?
8. What is nature of queries on OLAP database?
9. What are main components of Business Intelligence of a company?
10. Which of the following requests are typical for OLAP database:
    a. What was the increase in sales of product `X` during the weekends, if compared to increase in sales on weekdays for last two years?
    b. Sales of which products dropped this year more than 10% comparing to the last year sales?
    c. How many of product `X` was sold today in store `Y`?
    d. What was the price of product `X` yesterday?
    e. What was the average price of product `X` for different months of last year in the stores of `Y` sales region?

Practical Assignments

1. Describe benefits of merging tables `Subcategory`, `Category`, and `Product` into one dimension. Does this design have any disadvantages, e.g. increased size of the database or worse performance?
2. On dimensional model of `Sales` case write queries for analyses of sales for:
    a. Sales district `X` for last three years.
    b. Products of category `Y`.
    c. Sales region `X` and product category `Y`.
    d. Stores of district X and products of category `Y` for the previous year.
3. Merge tables `Product` and `Store` in one dimension. On this design, write queries from assignment 2 and explain disadvantages of your approach (increased size of the database, worse performance) as compared to the design discussed in the handout.
4. Assume that for `Sales` case you decided to have dimensions `Product`, `Subcategory`, `Category`, `Store`, `Region`, and `District`. Build `Fact` table for your approach. Write queries from assignment 2 for this design and explain its disadvantages (increased size of the database, worse performance) as compared to the design discussed in the handout.
5. Based on assignments 1, 3, and 4 formulate the basic rules for design of dimensions.
6. Based on the rules of design of dimensions formulated in assignment 5 explain how you will change the dimensional model of `Sales` case if users need to analyze the business in the context of:
    a. Diet type of products.
    b. Gender and age of customers.
7. Describe the regular structure of queries on OLAP database. Demonstrate this regularity on the queries of assignment 2.

References:

1. R. Kimball “The Data Warehouse Toolkit”, John Wiley & Sons, Inc., 1998.
2. C. Imhoff, N. Galemmo, J. Geiger “Mastering Data Warehouse Design”, John Wiley & Sons, Inc., 2003.
3. C. Adamson, M. Venerable “Data Warehouse Design Solutions”, John Wiley & Sons, Inc., 1998.
4. W. Inmon “Building the Data Warehouse”, John Wiley & Sons, Inc., 2002.