INTRODUCTION

Data warehouses (DW) appeared first in industry in the mid 1980s. When their impact on businesses and database practices became clear, a flurry of research took place in academia in the late 1980s and 1990s. However, the concept of DW still remains rooted on its practical origins. This entry describes the basic concepts behind a DW while keeping the discussion at an intuitive level. The entry is meant as an overview to complement more focused and detailed entries, and it assumes only familiarity with the relational data model and relational databases.

BACKGROUND

Databases in the 1970s and 1980s were used mainly to maintain data for everyday operations. A typical example is a bank database holding information about accounts that is used by a network of ATM machines. This is called operational data, and the role of the database is mainly to support transactions. Transactions are operations that access and change the data in the database; in the example of the bank, an ATM may access the database to check if a client has enough money in her/his account, and to change the balance if a withdrawal or deposit takes place. While transactions usually affect only small parts of the database, databases have to handle efficiently a large number of transactions, many times concurrently. Thus, the database architecture is geared towards efficient support of small, localized access. This is called on-line transaction processing, or OLTP. However, in the mid 1980s, emphasis shifted towards comprehensive analysis of current and historical data, in order to understand business patterns. This analysis is high level and involves many related low-level items and has as its goal to support decision making. Hence, this kind of analysis is called decision support (DS), but the term on-line analytical processing, or OLAP, is also used, to emphasize the differences with OLTP (Kimball & Strehlo, 1995). The typical DS task involves summarizing large amounts of low-level data and relating different aspects of the business to find interesting correlations, and therefore database access is based on complex queries. Because of the business intelligence it provides, OLAP grew rapidly and nowadays is a necessary tool for most medium and large size enterprises.

However, the analysis that DS relies upon is made complicated by a historical factor. In many organizations, especially those of a certain size and complexity, data was stored in several separated data sources, because each unit within the organization developed its own database to support its informational needs. Decision support requires that all the data in the different databases is put together to yield a global picture of the organization. As an example, imagine a hospital where the surgery unit has developed a database of patients undergoing surgery (with information on the type of surgery), Pharmacy has developed a database of patients that are administered some medication (with information on the dosage, time, etc.), and accounting has developed a database of patients in order to bill them. If the hospital’s management asks for an analysis that correlated patient’s socioeconomic status with type of surgery and amount of medication prescribed, the information needed is dispersed throughout all three databases. If patients are identified by social security number in one database, patient ID in another, and a combination of name and date of birth in another, putting the information together may be complicated. Thus, many companies decided to centralize and consolidate all their data in one central repository: a DW. In the following, we describe the main characteristics of a DW, its design and implementation.

DATA WAREHOUSING

Data Warehouse Design: The ETL Process

A DW contains a copy of the data from other databases, which are called the data sources of the warehouse. In order to get the data from those data sources (usually OLTP databases) and allow them to continue working normally, the data from the sources is copied inside the DW at regular, preestablished intervals, to refresh the DW. Because there are multiple data sources, it is necessary to watch out for redundant (duplicate) data,
missing data, or heterogeneous data. This is done during
the *extraction, Translation, and loading* (ETL) process
(Immon, 2002; Kimball & Ross, 2002).

The ETL process involves extraction, transformation,
integration, cleansing, loading and computation of
additional data (different authors give somewhat different
phases, or name them differently). Extraction refers
to the act of capturing the data from the sources and
physically sending it to the DW. Usually, standard ex-
ternal interfaces supported by the databases (called
gateways) are used. To make the process efficient, it is
desirable to keep track of which data is copied in the DW
on a given extraction, so that only new data is copied on
the next extraction.

Different databases (especially historical ones, or
legacy systems) may contain data which is represented
in different formats. Thus the *transformation* step,
which involves converting data to a uniform format,
removing, adding and reordering attributes if necessary
(e.g., adding a key, or a timestamp).

Because different databases were created by different
people for different purposes, it is likely that they contain
related (or overlapping) information stored in different
ways. When the data is integrated, there may be problems
of *homonyms* (i.e., the same name for different concepts),
synonyms (i.e., two names for the same concept), and many
other semantic mismatches. Thus, another step in extract-
ion is to *integrate* the data, that is, to merge and match data
to ensure that data about the same entity is integrated and
data about different entities is separated (our previous
hospital example provides an instance of such semantic
mismatches). A separate step is *data cleansing*, which
involves making sure that the data is as devoid of noise as
possible; typos, data entry errors, missing data are dealt
with at this stage.

Finally, the data must be loaded into the DW. This is
a complicated task because, in normal operating mode,
the DW is queried and the data is not changed to maxi-
mize performance answering queries. Thus, the data in
the DW is considered static and left to be out of sync for
the time between refreshes. Then, a large amount of data
must be loaded in the DW at once to refresh it (batch
load); this is called *DW refreshing*. The window of
opportunity depends on DW usage. At loading time,
other activities that will help improve query processing
must be carried out (e.g., sorting, summarizing, index-
ing, partitioning, checking integrity constraints). As a
result, loading the DW is a complex process in itself.
Because of large volumes and shrinking time windows,
most commercial utilities use *incremental* approaches,
in which old results are reused and only truly new data is
added (Jarke, Lenzeneri, Vassiliou, & Vassiliadis, 2000).

To manage the DW, a *metadata repository* is cre-
ated. This is a system catalog that contains metadata
(i.e., information about the data in the DW, including
their origin, format, intended meaning, range, and informa-
tion about the source of the data). The catalog is
usually large and complex, because it used to support all
steps in the ETL process. Thus, it is not surprising that
sometimes the metadata repository is kept in a database
itself, and that building it is one of the most difficult and
important steps in the process of building a DW.

Currently, ETL is often done off-line, by hand, and by
replicating almost everything. There are tools to help with
this task: data migration tools allow simple transformation
rules to be applied to the data (“replace string *gender* with
string *sex*”); *data scrubbing* tools use domain-specific
knowledge to do the cleansing (“ages should be between 18
and 65”). They often use parsing and fuzzy matching tech-
niques to add some intelligence to an otherwise complex
and laborious process. The tools that different systems
offer to handle ETL vary enormously from system to
system (IBM, 2004; Oracle, 2004).

**Data Warehouse Design:**
Logical and Physical Models

The basic DW design is influenced not by theories of
*normalization*, as in regular databases, but by the fact
that the DW goal is to provide information about a
business model.

The data in a DW is analyzed in terms of *facts* and
dimensions. A *fact* is a basic business datum, contains
raw data, and is irreducible (e.g., point-of-sale informa-
tion: who bought what, when, where and at what price
is the basic fact on a retail enterprise). A *dimension* is an
attribute of the fact (e.g., Client, Product, Time, and
Store are dimensions of the point-of-sale fact). Usually,
there are several dimensions to every fact. Each
dimension can have in turn a set of associated attributes.
For instance, Store may have attributes Name, City, and
State associated with it. It is also typical that the values
of a dimension can be organized in a hierarchy. The
typical example is the dimension *Time*, which can be
organized in Date, Week, Month, Quarter and Year. The
previous example, the dimension Store, can also be
organized in a geographical hierarchy through City and
State. Finally, besides dimensions, a fact contains *mea-
sures*, usually numerical values indicating some proper-
ties of the fact. For instance, in our point-of-sale ex-
ample, Price, Quantity and Amount may be three dimen-
sions. Obviously, these definitions are informal; what is
fact and what is a dimension depends on the DW subject
and on the business analysis that the DW is to support
(hence the usual statement that DW design is subject
oriented).

The basic data model choices are relational OLAP
(ROLAP) and multidimensional OLAP (MOLAP).