# Database Basics

## 1. Definitions

### 1.1. Data mining

→ **Finding new information**

### 1.2. Data warehouse

*"A data warehouse is a single source for key, corporate information needed to enable business decisions."*

→ **Using available information**

## 2. Data

### 2.1. Classifications

Several separated data sources contain crucial information (information that is important to a company). They must be integrated in order to get the full picture. An integration will create an integrated system (IS), but not necessarily a data warehouse.

There is two major ways of integrating information, a logical (virtual) approach and a materialistic (physical) approach.

**Virtual** can be compared to creating a view over database tables. Views always contain the newest information. In order to access it the data might sometimes be denormalized (when using joins for example), so there might be a high latency. As you only display data you can easily modify (insert, update, delete) and maintain data in the sources without affecting the whole system.

**Physical** can be compared to creating a temporary table that contains all denormalized data from a view. It is a copy, so therefore the data is always as old as the date of the last copying process. When accessing the information the data is most likely to be denormalized already so there will be a lower latency. The data is denormalized and therefore hard to maintain.

| | Virtual | Physical |
|---|---|---|
| Actuality | High | Low |
| Latency | High | Low |
| Flexibilit | Maintenance in Source | Maintenance in IS |

These are not data warehouse classifications but classifications of integrated systems in general.

A data warehouse is a persistent physical copy of data of all data sources, so it is built from a materialistic architecture. Other materialisic IS are operational datastores. Important virtual IS are "federated" or "mediator-based IS".

# 3. Techniques

## *3.1. Extract, Transform, Load*

The ETL is an easy but precise concept. Extraction means accessing data from a data source (eg. an excel sheet, a binary file, a database, a website, or whatever ), then cleaning it up and transforming it to the target data storage, where it will be loaded into a persistent format. ETL is important for data warehouses.

Each step is a logical step. There is no assumption if it is program, a function within a program, a process, a script or whatever. What matters is *what it does*, not how it works. So the **Extraction** can be anything you can think of. Write it in your favorite programming language.

### *E - Extraction*

The data from the data source will be accessed. For this it will be opened somehow. In most cases this means opening a file or establishing a connection to the database server. Then some sort of program logic will be executed in order to read the contents of the file. For this you will have to look out for the syntactic properties of the file such as the string delimiter in a Comma Separated Values file (csv).

### *T* - Transformation

In the transformation step the main work is performed. In transformation you clean data, you convert it, you change it, recalculate it, etc.. It's totally up to you.

In order to perform a transformation you will need a mapping instruction. That is some sort of table that contains information about the input data and the output data. This is needed to know what to transform at all.

***As an example:*** You have an excel sheet and open it. In there is a list of strings. The program then requires information about what column from the excel sheet should go in what table column. Also it must know the source and target type.

## *L - Load*

The data warehouse (or data target of some sort) loads the new data. It will keep a history of the whole process so the whole process can be reverted and evaluated later if something goes wrong with the data.

### *3.2. Applying ETL*

As stated above ETL is a concept. You figure out what to do. During the lecture we will have the example to import an excel sheet into a database table. Here we used Microsoft SQL Server Integration Services. You could have used any other program for that (that might have been faster but without the fancy diagrams and all). What we are going to do is open the excel file (E), use a C# script to parse the values from the strings (T) and then saved it in the new database table (L).

This is actually not big science - its common sense. Daily stuff! There is a million ways to do stuff like that but these are the three most general steps.

Not only will you write your ETL application, run it and throw it away, you will periodically recall the application to restock your data warehouse. The whole extraction can be done periodically (nightly, weekly (weekendly), monthly …). What interval you choose is not fully up to you. In most cases you will perform copy operations when the data sources are not in use or at least not at 100% peak. For most companies this is at nighttime, because most customers are asleep. So every night most companies mirror their data into their data warehouses and also to other locations where they need the data.

Not only can these things be called periodically but also triggered by events or explicit request. A request is a user clicking some sort of "Extract" button in an admin interface. A trigger is a program in a data source that reacts to a certain event, such as insertion of new records or reaching the limit of a certain quota. To make an example this could be a trigger that reacts to insertions on a table. If there are more than 1000 customer requests in the table he copies them to the data warehouse.

So remember, data warehouse ETL periodically reoccurs.

### 3.3. More about Transformation

The main part of ETL is the transformation part, as accessing tables, files or databases isn't that big of a deal. The majority of work is done during transformation. It is what usually takes longest.

During transformation you will do a semantic analysis of the values of the data source. You will now recalculate columns and perform conversions. Common conversions are type conversions such as parsing integers out of provided strings. Also you will change the cultural dependencies such as units and measurements (Euros to Dollars, Centimeters to Inches). To exactly know what comes in and what must go out you will need a mapping table.

One major part of transformation - besides the conversions - is the cleanup process. Not only will you convert a table into another but you will merge different tables, join them, slice them and whatever is required. The source and the result never look the same. To do so you will change the normalization of the input and output data to fit the desired target structure.

In most cases you will try to normalize the input data into a special dimension tables within a star-schema (see: http://de.wikipedia.org/wiki/Sternschema) by doing that you can perform analytical operations on the data target (the data warehouse). These operations are called OLAP, which is another set of approaches and will be covered later.

The main part of the normalization is finding duplicate entries within the input data. The normalized dataset is not supposed to contain redundant data (in order to be normalized), so they have to be stripped out of the working data. Finding duplicates is pretty complex and also requires fairly expensive operations (expensive on the resources). The overall process of finding duplicates is called **object identification.**

### 3.4. Object Identification

**OI** is used to resolve redundancies within the database by handling duplicates that appear in a set. You need to perform several tasks in order to remove duplicates. First, you need to have the data as a workable set, which means that at least the equivalent data types should be fit. If a value is 5 it should be an *Int*, not a *tring*. After that, you can partition the data. That means that you split the data into a pre-normalized form, so it is separated as good as possible, but not necessarily with relationships.
There are identical and non-identical duplicates.

After that you can apply algorithms. These algorithms are partially highly mathematical and hard to learn, but you find resources for that all over the internet. The most important algorithms are:

- Levenshtein Distance
- Hamming Distance
- Typewriter Distance
- Sorted Neighborhood ("Sortierte Nachbarschaft")
- Phonetic Algorithms
- Soundex
- Kölner Phonetic