# Unit 03 Notes

## Data Constructs Overview

Our previous lecture presented the origins of key terms such as data warehouse, business intelligence, and data mart. We mentioned that they refer today to components of complex information ecosystems with architectures such as the Corporate Information Factory (CIF). We also listed the main CIF components and their roles. This lecture provides more precise and current descriptions of these terms, as well as the differences between them. We also present the CIF architecture in more detail.

The CIF takes all raw transaction data from the **External World** (chapter 3, optional reading), and produces information for consumers that are also in the External World. Therefore, the characteristics of the External World (such as the taxonomy of transactions that must be handled, and the type of information consumers that must be served) are fundamental design specifications for the CIF.

The CIF collects the External World data through the **Applications** component (chapter 4, optional reading). Applications interact directly with the data producers, gather raw data from them, and edit them for quality. Many applications were not designed to work as parts of a CIF. They are older, slower, and difficult to change or replace. In addition, they are not integrated; each application works with a separate data producer. Therefore, the data produced by the application component is not integrated either; it may have duplicates and inconsistencies in keys, definitions, encoding, and other features.

Some people wonder why copy the data in a data warehouse rather than use them directly from applications. Others wonder why not integrate in applications rather than introduce an additional component. Both approaches would be very inefficient in large enterprises. The HR may be supported by an ERP system such as SAP, the training processes may be supported by a third party system, while the point of sale transactions could be stored in an Oracle DB. To assess the impact of training on sales reps' performance, we need to query multiple systems, each with their own schemas and granularity. In addition, querying makes use of business rules, which need not be known by the users of the application component. It emerges that doing the querying and analytics in applications would be very inefficient. Applications interact directly with customers and should be as responsive as possible. That is why integration and analytics should be implemented in other CIF components.

Data integration is the main task of the component that takes the data produced by the applications. This component is appropriately named the **Integration and Transformation** (**I&T**) Layer (Chapter 5). It consists mainly of programs, unlike the ODS, DW, and DM, which are mostly data. The programs produce the integrated data according to the data model. The logic of transformations undergone by the data is recorded by the I&T layer as metadata. This recording of metadata over time is called **versioning**. The I&T Layer also selects the **system of record**, i.e., the best source of application data that will be processed further. Chapter 5 outlines several types of data processing done by the I&T Layer. These transformations turn the raw data into a corporate asset that is ready to feed the ODS and the DW components. The data feeds into and out of the I&T Layer are depicted in Figure 5.5 (pp 66).

An **Operational Data Store** (ODS) is a collection of data supports both operational processing and decision making. It has the following properties:

- **subject-oriented**: organized according to subjects of interest for querying and analytics (e.g., customers, products, vendors, orders, accounts);
- **integrated**: all data is consolidated by the I&T Layer, to use consistent naming conventions, definitions, key structures, encoding, and so on;
- **volatile**: it is updated during normal processing;
- **current-valued**: the data is daily, weekly, or monthly, but not older;
- **detailed**: it only contains detailed data, not summary data.

The ODS is difficult to build and use because it must support both operational processing and decision making, which means that it must support all types of data accesses. This makes it more difficult for developers to optimize the ODS for specific functions without negatively impacting other functions. The data feeds into and out of the ODS are presented in Figure 6.2 (pp 82).

A **Data Warehouse** (DW) is a collection of data that supports reporting and analytics. It is subject-oriented and integrated like the ODS, but the remaining three properties are reversed:

- **non-volatile**: the data remain static, or read-only. New data are periodically appended, but never changed, except for correcting errors;
- **time-variant**: any record in a data warehouse is accurate at a specific moment of time, and no inference should be made regarding its accuracy at different moments of time. Basically, a data warehouse contains a series of data snapshots that can span over 10-20 years;
- both **detailed** and **summarized**: it contains both detailed and summary data.

The DW is often one order of magnitude larger than any other CIF component in terms of data storage and processing. The growing volume of DW data is bound to strain the DB capabilities at some point. For this reason, the DW data is regularly archived on the **Alternative Storage Component** (chapter 10, optional reading). The data feeds into and out of the DW are depicted in Figure 7.3 (pp 100).

Inmon's variety of data warehouse stores **normalized** data. Such data allow only a limited amount of reporting and analytics. The rest takes place in the data marts, which are **denormalized**. Denormalization is aimed at reducing the need for **relational join** operations when the data warehouse is queried for decision support. In contrast, Kimball's variety of data warehouse is dimensional, i.e., it divides the data into facts and context (**dimensions**). Both approaches have pros and cons.

Data size has a strong impact on the design of DW, ODS, and I&T. Small companies might not need an ODS at all, and might implement the I&T layer as simple file transfer and basic business rules. Such a small DW is similar to a data mart. Very large companies, on the other hand, might require a federated query layer.

The following table compares the main ODS and DW characteristics:

| Characteristic | Operational Data Store | Data warehouse |
|---|---|---|
| **Primary purpose** | Run the business on a current basis | Support managerial decision making |
| **Design goal** | Performance throughput, availability | Easy reporting and analytics |
| **Primary users** | Clerks, salespersons, administrators | Managers, business analysts, customers |
| **Subject-oriented** | Yes | Yes |
| **Integrated** | Yes | Yes |
| **Detailed data** | Yes | Yes |
| **Summary data** | No | Yes |
| **Time of data** | Current state of business (day-week) | Historical snapshots (mo-yr), predictions |
| **Updates** | Frequent small updates | Periodic batch updates |
| **Queries** | Simple queries on few rows | Complex queries on many rows |

Some companies build one or more **data marts** (Chapter 8) to address the reporting and analytical needs of specific departments. A data mart contains a **denormalized** subset of the DW data and much summary data. Since they only need to support a smaller set of analytics, data marts can be deployed faster and at smaller costs than full data warehouses. Data marts can even be deployed without a full data warehouse, but there are disadvantages to that approach. This topic is subject to an architectural debate that will be addressed in more detail in Week 13.

For reasons similar to the ones for building data marts, CIF developers may build an **Exploration Data Warehouse** and a **Data Mining Warehouse** (Chapter 9, optional reading). As their names imply, the former is tuned to the needs of **explorers**, who have very large and hard to predict queries, while the latter is build specifically for **miners**, who engage in data mining activities. The "regular" data warehouse can thus focus on **farmers**. The lesson is that we can spawn copies of a data warehouse, optimizing each copy for a different type of users. A different kind of data warehouse multiplication occurs in global enterprises or national governments, which deploy information systems with **multiple data warehouses** (Chapter 17, optional reading).

## Data Warehouse Components

A data warehouse makes use of at least two storage areas: a **source area** and a **target area**. The source is the system of record that stores the operational transaction data. Sources are heterogeneous, and might include ERP systems, relational databases, and legacy systems. The term "source" (for a component ABC) is used in a CIF to refer to whatever component feeds the data to component ABC. The target area hosts the final data assets for review after all transformations have taken place. If the sources are not integrated, the data warehouse developers must use **ETL** (**Extract**, **Transform**, **Load**) tools to process the raw data. These tools a type of data storage termed **staging area**, which is a temporary storage area (database) used for data transformation and cleansing. The data in the staging area is continuously changing, thus it's not good for querying. It can be erased after successfully loading of the DW. In addition to storage areas, a data warehouse also consists of **subjects** of **interest**, **business rules** applied to raw data to produce corporate data, as well as **data types** and **granularity**.

# Target Industries for Data Warehouse Deployment

The conceptual roots of data warehouses are older than of modern computers. Two pioneers in **marketing research** worked diligently to integrate quality historical data for business analytics. One is Charles Coolidge Parlin (1872-1942), the father of marketing research. He gathered information about customers and markets to help Curtis Publishing sell more advertising in their magazine, The Saturday Evening Post. The other pioneer is Arthur Nielsen (1897-1980), the founder of AC Nielsen. He made many innovations in **target marketing** and media research, and gave practical meaning to the concept of market share as a critical measure of corporate performance.

**Target marketing** has been a fertile ground for data warehouses ever since. Companies want to understand individual customer demographics, behaviors, and product needs. A data warehouse enables the "demassification" of the market, i.e., customer segmentation based on historical data. Historical data are also used to develop response models that assign probabilities to an individual or group responding to a particular type of marketing campaign. Better targeting increases response rates, yields better customer retention and acquisition, as well as more cross sells.

**Customer profitability analysis** has also benefited from successful data warehouse deployments. Using cost models based on customer activity and detailed revenue reconciliation, a company can assign a profitability score to each customer. This enables the identification of profitable customers, as well as the non-profitable ones. The company can then use adequate strategies for both categories. In many cases, 80% of a company's profit comes from less than 20% of its customers (the 80/20 rule).

**Customer retention** is another traditional data warehouse application. Acquiring a new customer may cost five times as much as retaining a customer, and a retained customer can generate twice the income of a new customer. Historical data captures signals about customer intentions long before customers act upon them. At-risk scoring models can make predictions based on these intentions, and the company can start customer retention programs.

Target marketing, customer profitability analysis and customer retention are areas of **Customer Relationship Management** (CRM), a field that combines knowledge management, data mining, and data warehouses. A DW built for CRM is sometimes termed Customer Data Repository (CDR); there are no standard rules for CDR design. It is estimated that more than 50% of CRM solutions fail because of inappropriate processes or poor selection of technologies.

**Fraud detection** is a popular DW application because it impacts the bottom line in many types of businesses such as credit cards, phone cards, insurance claim processing, merchandise returns, airline ticketing, and financial transactions. Fraud detection models are trained on DW historical data and deployed on the operational systems, with some key historical variables fed by the DW.

Finally, DWs have also been successfully used in **credit risk analysis**. Major creditors employ quantitative models of credit payment behavior based on individual's history. These models combine internal data warehouse data with external credit bureau data about customers to assess their credit worthiness. Accurate models offer a significant competitive advantage.

# Building the Data Warehouse: the Inmon method

In the development of a data warehouse, Inmon argues that the data are available from the beginning, whereas the business requirements are not. Thus, a requirement-driven development cycle, such as the "**Waterfall"**, is not appropriate for DW development. A data-driven development approach for the data warehouse is proposed instead. We start the development by integrating some operational data and implementing very basic analytic capabilities. These attract a few users and help us understand what we can actually accomplish with those data. Then we integrate more data and develop more analytics to attract more users and get even more opinions and requests. Then we just keep improving the system using this iterative approach, usually termed a "**Spiral"**. Inmon calls it "**Day 1 to Day n Phenomenon**", with the implicit understanding that each "day" can span over weeks or months:

Day 1: The DW developers get acquainted with the legacy systems doing transactional processing

Day 2: They build and populate the first few tables of the first subject area. The users take a look at these tables and begin to notice their analytic capabilities.

Day 3: More of the DW is populated and more users are interested in the integrated source of data. The first serious DSS analyst gets involved.

Day 4: Even more of the DW is populated. The DW becomes a source for analytics. Many DSS applications spring up. So many users compete for DW access, that some of them get put off by the effort.

Day 5: Departmental DBs (data marts or OLAP) appear. They bring data from the DW in their own environment and summarize it for their own needs. More DSS analysts are attracted.

Day 6: Land rush to department systems. DW end users begin using them rather than the DW.

Day n: The DSS based on the DW is fully developed. The DW is full of data, but few end-users access it directly. Most use data marts instead. Most analytical processing occurs there too.

The end result of this iterative development process is a data warehouse environment with one or more layers of data marts, each layer using a different data granularity. Details on this process are available in Inmon's Building the Data Warehouse book, which is the "how" book of Inmon's data warehousing vision (the CIF book that serves as our textbook is the "what" book).

We now switch to Kimball's data warehousing vision for the next several lectures. It is important to notice, however, that a data warehouse is, to a certain degree, decoupled from its development process.