8 Databases, Big Data and Business Intelligence

(Draft: June 15, 2016)

Opening Vignette: How an Electronic Health Record Saved a Patient's Life

Electronic Health Record (EHR) is a database of patient personal (e.g., age, weight, smoking habits) and medical information (e.g., blood pressure, allergies. previous operations, current medications). This database serves as a central repository that supports clinical decisions and many other patient care applications.

In this story, Larry Garber, an internist at Worcester, Massachusetts-based Reliant Medical Group, explained how a simple decision support system based on EHR saved the life of one of Reliant's patients. A notification on the screen of Reliant's patient portal alerted a 65-year-old Medicare patient that he was due for the Abdominal Aortic Aneurysm (AAA) screening. The notification for the AAA screening is triggered when men who smoked in the past turn 65. The patient contacted the hospital because he saw the reminder which was triggered by his prior smoking history in his EHR. A screening ultrasound was scheduled and a large aneurysm was found. If the aneurysm had ruptured, it might have killed the patient. The patient was promptly admitted to the hospital and had the aneurysm repaired.

EHRs are at the core of modern healthcare systems and provide many benefits to the patients. They not only keep track of patient records but also can be used to detect and predict the type of preventive measures that are needed. This can result in reminders and warnings for the patients that could save their lives.

Source: Coughlin, B., "How a screening prompted by clinical decision support saved a patient's life", https://www.healthit.gov/buzz-blog/ehr-case-studies/screening-prompted-clinical-decision-support-saved-patients-life/, accessed June 6, 2016.

8.1 Introduction

Data, typically stored in databases and other data stores, are used to make business decisions and support almost all business applications. The data stores may use different data management systems from different vendors and may reside on different computing platforms which are interconnected through different networks. The purpose of this chapter is primarily to serve as a tutorial on how data is stored, organized and used for different applications in modern enterprises. Special emphasis is placed on the recent trends in Big Data and business intelligence. Specifically, we attempt to answer the following questions:

- What are the basic data concepts
- What is enterprise (corporate) data and how it is evolving to Big Data
- What are Database Management System (DBMS)
- What are relational databases and what is SQL
- What are the basic principles of database design
- What are object-oriented and NoSQL databases
- What are data warehouses and how can they be used
- What is Hadoop and why is it needed
- What are the decision support and business intelligence (BI) tools and techniques
- What is data mining and how is it used to make management decisions
- What is the main idea of data analytics and deep learning

Importance of Customer Information: An Example

The following example of customer information was introduced in another Module and is based on the article "Cultivating An Information Culture: An Interview With Thomas Davenport", CIO, January 1995. Let us revisit this example from a data warehousing perspective.

An airline was flying a customer to Europe on business trips. The customer was not happy because he had lost his luggage a few times and the flights were typically late. So the customer stopped using the airline.

The airline did a lifetime profitability analysis of this customer and found that the airline had lost a potential $400,000 over the life of the customer (had he stayed with the airline). The airline also found out that there were several clues about his unhappiness (he had called the lost luggage department several times and had written complaint letters to the chairman).

Instead of developing a composite and complete profile of the customer history, each incident was treated separately and the customer was sent two free-drink coupons every time he complained.

What are the data warehousing implications of building a composite and complete customer information to avoid such situations. Here are the main ideas:

- A comprehensive data model with attributes to represent complete view of customer activities is needed.

- Access to information from different sources (luggage department, complaints department, flight information, etc.) is needed. Some of this information may be extracted and loaded into the data warehouse while other may be just accessed from back-end systems (i.e., "virtual data warehouse").
- Powerful analysis tools including data mining are needed for studying the customer activities and cross relationships.
- Top management involvement is essential because individual departments may claim ownership of data (e.g., flight information).

8.2 Concepts and Definitions

8.2.1 Data Definitions

Computer systems organize data in a hierarchy that starts with the bit (abbreviation of *binary digit*), which represents either a 0 or a 1. Bits can be grouped to form a byte to represent one character, number, or symbol. Bytes can be grouped to form a field, and related fields can be grouped to form a record. Related records can be collected to form a file, and related files can be organized into a database. Figure 8-2 shows this data hierarchy.

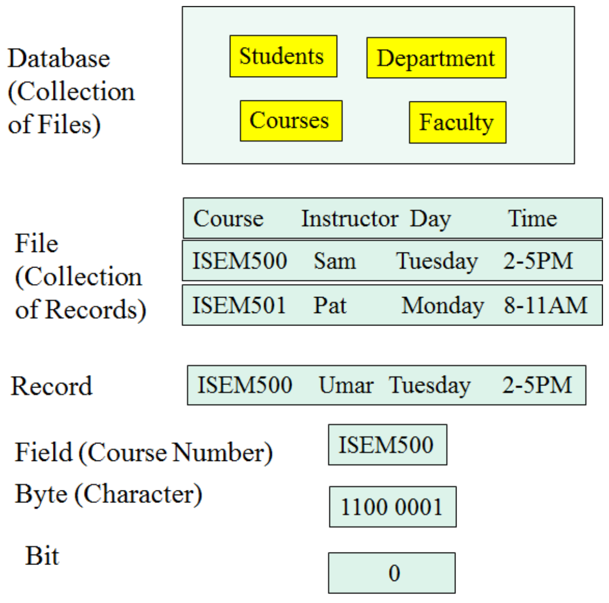
Database (Collection of Files)
Students
Department
Courses
Faculty
File (Collection of Records)
Course Instructor Day Time
ISEM500 Sam Tuesday 2-5PM
ISEM501 Pat Monday 8-11AM
Record
ISEM500 Umar Tuesday 2-5PM
Field (Course Number)
ISEM500
Byte (Character)
1100 0001
Bit
0

Figure 8-1: Data Hierarchy

- Field: Group of characters as word(s) or number
- Record: Group of related fields
- File: Group of records of same type

- Database: Group of related files
- Record: Describes an entity
- Entity: Person, place, thing on which we store information
- Attribute: Each characteristic, or quality, describing entity, e.g., Attributes Date or Grade belong to entity COURSE

8.2.2 Enterprise (Corporate) Data

Enterprise Data, the data that is shared by different business processes, is a cornerstone of many applications and services in contemporary environments. This data (also known as Corporate Data) is the information that is used or created by an enterprise in conducting business and is shared across the business processes of an enterprise. As shown in Figure 8-2, this data consists of information that is shared by several applications and is also the main source for data mining and business intelligence. Specifically, the shared corporate data consists of information from almost all corporate resources (e.g., customers, products, partners, finances and employees) in different shapes and forms that is used in business decision making through simple queries to sophisticated data mining and analytics. In essence, shared corporate data supports an "ERP of ERPs" as shown in in Figure 8-2.

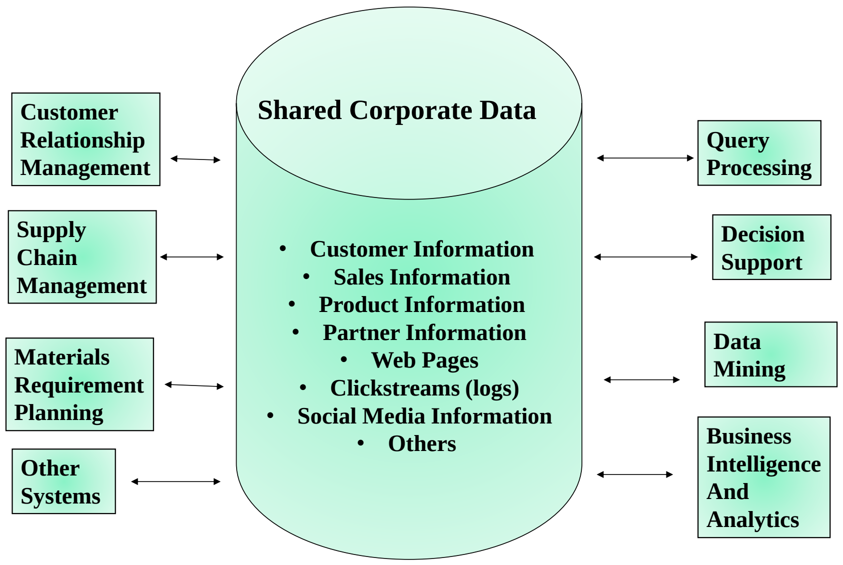

Figure 8-2: Conceptual View of Shared Corporate Data – An ERP of ERP View

A sim[le example of shared enterprise data is customer information that is used in different applications such as order processing, billing, accounts receivable/payable, and marketing information systems. From a new application development perspective, the needed data must be "architected" (i.e., modeled, designed, allocated, interconnected) and managed carefully, even though the application functionality could be satisfied by an off-the-shelf package.

Although it is possible to discuss the enterprise data issues from a single application's point of view, it is essential to discourage this view. Why? This is because the enterprise data is shared by multiple applications and end-user tools over a very long time frame. The enterprise data; thus, must not be viewed as serving one application — enterprise data is an enterprise asset for ERP of ERP purposes and is not a single application or ERP captive property. This view, particularly true for decision support and knowledge management applications, suggests that an enterprise data architecture needs to be established before diving into the individual application architectures. In many practical situations, however, software and data architectures are developed in parallel as part of solution architectures.

8.2.3 Big Data – Evolution of Enterprise Data

8.2.3.1 What is Big Data

Big Data is a large collection of diverse datasets that cannot be processed using traditional computing techniques. The basic reason for paying attention to Big Data is that the generation and use of data has exploded since 2010 due to the advent of new technologies such as sensors, mobile devices, and communication means like social networking sites. As observed in the side bar "The Data Explosion", almost 90% of the world's data was generated in the last few years. At present, the term "Big Data" refers to the physical size of the data but also to the associated tools such as analytics.

The Data Explosion – Big Data

We have been making decisions based on data since the start of time. For example, the Pyramids in the ancient Egypt were constructed by using detailed designs and calculations based on whatever data was available at that time and the Roman Empire built roads between different regions of the Empire by using geographical data.

However, the generation and use of data has exploded since 2010 due to the advent of new technologies such as sensors, mobile devices, and communication means like social networking sites. The following statements, based on a SAS Tutorial, capture how rapidly the amount of data produced by mankind is growing:

- The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes
- The same amount was created in every two days in 2011
- The same amount was created in every ten minutes in 2013
- 90% of the world's data was generated in the last few years

This data explosion has resulted in numerous new terms such as "Big Data", topics to study such as "Data Sciences", and tools such as analytics. We will review many of these topics in this chapter.

There are several well known examples of the Big Data such as the World Bank Open Data Initiative. Similarly, the United Nations, the World Health Organization (WHO), the World Economic Forum (WEF), and many other organizations are making tremendous amount of data openly available to private as well public enterprises (see the sidebar on "Open Big Data" for more details). Shared Corporate Data has become part of Big Data because Big Data adds *external data (e.g., data available through government agencies)* to the Shared Corporate Data for improved decision making. This data significantly complements and augments the Shared Corporate Data of enterprises for extensive analysis. Big Data is typically characterized by three "V" parameters:

- **Volume:** number of data items are extremely large

- **Variance**: the data format is highly variant – instead of highly structured data suitable for SQL queries, it may represent voice, images, and data captured through sensors
- **Velocity**: the data accumulates extremely quickly – for example, millions of images can be sent by a satellite in an hour.

Open BIG Data: Samples of Free Sources of Big Data

The following selected sources of Big Data contain free and openly accessible data at global levels. This data is not limited to one country or one subject area.

- World Bank Open Data initiative (http://datacatalog.worldbank.org/)
- UN Statistics Department (UNSD) Data Sources: UN Data website (http://data.un.org/) has 25 databases on topics such as crime, food, health and tourism. Also shows country wide statistics (an API is available for program access).
- UN Public Administration Network Data (www.Unpan.org)
- The Organization for Economic Co-operation and Development (OECD) has a very large statistical data, available at http://stats.oecd.org/
- National Climatic Data Center : Huge collection of environmental, meteorological and climate data sets from the US National Climatic Data Center. The world's largest archive of weather data. http://www.ncdc.noaa.gov/data-access/quick-links#loc-clim
- Data.gov: http://data.gov. The US Government pledged last year to make all government data available freely online. This site is the first stage and acts as a portal to all sorts of amazing information on everything from climate to crime.
- US Census bureau: http://www.census.gov/data.html. A wealth of information on the lives of US citizens covering population data, geographic data and education.
- European Union Data portal: http://open-data.europa.eu/en/data/. Based on data from European Union institutions.
- US Health Care Data: https://www.healthdata.gov/. 125 years of US healthcare data including claim-level Medicare data, epidemiology and population statistics.
- National Climatic Data Center: http://www.ncdc.noaa.gov/data-access/quick-links#loc-clim. Huge collection of environmental, meteorological and climate data sets from the US National Climatic Data Center. The world's largest archive of weather data.
- Data.gov.uk: http://data.gov.uk/. Data from the UK Government, including the British National Bibliography – metadata on all UK books and publications since 1950.
- data.gov. One of the largest Big-Data sources available, with over 124,604 datasets available.
- Healthdata.gov. A US government provided data with large amount of data available for the healthcare domain.

8.2.3.2 What Does Big Data Consist of -- The Variance Dimension

Figure 8-3 shows a small sample of the variance dimension of Big Data – it displays what the Big Data consists of. Basically, Big Data resides on a variety of platforms (handsets, PCs, UNIX, mainframes) in a variety of formats (emails, word processing files, PowerPoint files, spreadsheets, social media messages, relational databases, catalogs, IMS databases, object-oriented databases, HTML documents, XML documents, indexed files, video clips, sensor data, satellite images, stock exchange data, power grid data, or design diagrams). All different types of data can be classified into:

- Structured data (e.g., relational data)
- Semi Structured data (e.g., XML data)
- Unstructured data (e.g., emails, Word, PDF, Text, Media Logs, satellite images, etc)

Although a large portion of enterprise data is stored in relational databases, Web data in the form of HTML/XML documents and "clickstreams" (logs of users clicking on HTML pages) is growing dramatically. In addition, data from social media, video clips, and satellite images is increasing rapidly. The Big Data is accessed by end-user tools (e.g., Web browsers, spreadsheets, mobile apps, SQL queries, data browsers, report writers, object-oriented viewers) from different suppliers and/or applications written in Java, C, C++, C#, Python, and other programming languages.

An interesting area of work, within the umbrella of Big Data is *Micro Data*. This data is being captured by millions on sensors that are being used a large number of mobile apps. A major area of work in Micro Data is related to IoTs (Internet of Things) that capture data about *everything* that is Internet enabled.

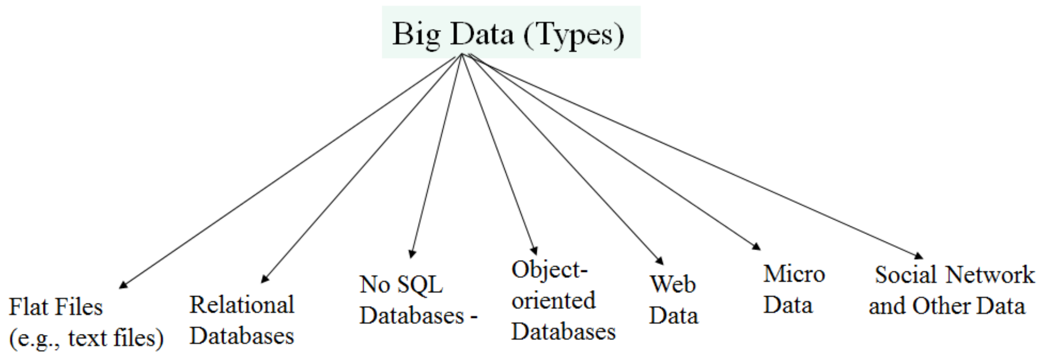
Big Data (Types)
Flat Files (e.g., text files)
Relational Databases
No SQL Databases -
Object-oriented Databases
Web Data
Micro Data
Social Network and Other Data

Figure 8-3: Big Data Components

8.2.3.3 Benefits and Challenges of Big Data

Big Data contains critical information that can be very valuable to our work and daily life. Here are examples of some benefits:

- Analyzing the data regarding the previous medical history of patients, hospitals can provide better and quick service.
- Understanding the information contained in online purchasing logs can reveal why some customers do or do not purchase products
- Reviewing the information kept in social networks is a very good tool for homeland security and also for marketing campaigns

- Combining data from multiple sources such as customer locations and GIS maps is very valuable for transportation companies

However, Big Data presents several challenges. The main challenge is that the data is too large and too complex to be processed by traditional computing approaches. A large number of tools and techniques such as the following are needed:

- Data storage and access technologies such as NoSQL databases are needed to handle unstructured data
- Sophisticated data analytics tools such as the one provided by Cognos and SAS
- Big Data Warehousing tools such as the Hadoop Framework

We will review some of these tools later in this chapter

8.2.4 Decision Support and Business Intelligence (BI) Tools

Data access and analysis tools for Corporate and Big Data have been known as decision support tools since the 1970s. However, business intelligence (BI) has become a more popular umbrella term since it was coined by the Gartner Group in the mid-1990s. In general, the term BI is used to describe a set of tools and techniques for the acquisition and transformation of raw data into meaningful information to support business decisions. BI tools provide historical, current and predictive views on data that is housed in different data sources, in different formats and at different locations. These tools fall into the following general categories.

- Data Query and Reporting Tools that provide a wide range of capabilities such as ad hoc query processors, point and click formulation of SQL statements, and Query by Example (QBE) that display tables that the user can fill in multiple rows with examples of the desired data.
- Visualization and Presentation Tools that present user or business oriented views of data (e.g., customer objects, population distributions and use statistics), dashboards, charts and graphs
- Executive Information Systems (EIS) that provide analysis capabilities primarily for the executive decisions. A typical example of EIS is "drill-down" tools that allow executives to start at summary information and successively look at the details that make up the summary information.
- Online Analytical Processing (OLAP) Tools emphasize the analytical aspects of decision support tools and are optimized for dynamic, multidimensional analysis of enterprise data for analytical functions such as forecasting.
- Data Mining Tools exploit a combination of AI and statistical analysis to discover information that is hidden or not apparent through typical query and analysis tools. Data mining tools are essential for Big Data because they focus on *discovery of information that is hidden in the massive amount of data.*
- Analytics Tools heavily rely on statistical analysis, quantitative methods, and mathematical models to help managers gain improved insights about their business operations and make better decisions. Data analytics of Big Data is a rapidly evolving area of work especially for business intelligence.

Many applications of BI tools can be found in health, education, public safety, public welfare, agriculture, transportation, tourism, and other public as well as private sectors. See, for example, the sidebar “BI Applications in Healthcare – A Quick Summary”. We will take a closer look at these tools later with particular emphasis on data mining and analytics.

BI Applications in Healthcare – A Quick Summary

Healthcare generates a tremendous amount of data during its financial, operational and clinical processes at various hospitals, claim processing centers, insurance companies and labs. BI tools gather the needed data from different sources, analyze it, and present it to different parties for decisions in disease management, clinical performance, process improvement, cost and waste reduction, and quality improvement. A few examples of BI use in healthcare are:

- Infectious disease prevention and controls by using the data collected through microbes, These sensors monitor the migrating populations and changing environments and send data to central data warehouses for analysis.
- Healthcare quality improvements through integration of patient health information from various healthcare providers. Patients can access and analyze their consolidated health information to self- manage their conditions.
- Diverse clinical specialties and scenarios can be supported through customizable BI tools. Some of these tools can be customized to handle outbreak of diseases like a virus, swine flu, Ebola, etc.
- BI tools such as dashboards, online analytical processing (OLAP), data warehouses, and data mining systems extract important business information from data on patient details, drug information, virus classification, clinical health etc.
- Hospitals and insurance providers use BI for their competitive advantage. For example, dashboards are commonly used to show cost reductions and improvements in patient satisfaction.

8.3 Data and Database Management Concepts

8.3.1 Files and Databases

At the lowest level, a *data item* is the smallest unit of data which cannot be subdivided. Examples of data items are part number, part name, weight, cost, etc. A *data record* is a collection of data items. An example of a data record is a part record which is a collection of part number, part name, weight and cost of a part. A *file* is a collection of similar data records. For example, a customer file consists of customer records and a parts-file consists of part records. Examples of files, also called “flat files”, are text files, html files, XML files, etc.

Conceptually, a *database* is a collection of files (dissimilar records). For example, a manufacturing database is a collection of data records and files associated with manufacturing activities (e.g., finished goods inventory, bill of materials, equipment), and a financial database consists of payroll data, accounts receivable, general ledger, etc. It is common to assign additional properties to a database definition. For example, according to Elmasri and Navathe [Elmasri 1989]: "A database has the following implicit properties:

- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot be referred to as a database.
- A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.
- A database represents some aspect of the real world. Changes to the mini-world are reflected in the database."

In a database environment, different users can view, access and manipulate the data in a database. The database may be a business database for financial and administrative applications; a manufacturing database containing bills of material, goods inventory and scheduling data; an office database consisting of memos and proposals; and a knowledgebase for artificial intelligence and expert system applications.

8.3.2 Database Management System (DBMS)

Database access and manipulation are controlled by a database management system (DBMS). A DBMS, shown in

Figure 8-4, is a software package which is designed to

- Manage logical views of data so that different users can access and manipulate the data without having to know the physical representation of data
- Manage concurrent access to data by multiple users, enforcing logical isolation of transactions
- Enforce security to allow access to authorized users only
- Provide integrity controls and backup/recovery of a database.

These functions of a DBMS are described later. As shown in

Figure 8-4, a typical database management system (DBMS) uses a database dictionary/directory to store the data views, data relationships, data formats and security restrictions; database logs to record the activities of transactions; and lock tables to allow synchronous concurrent access to the database by several users.

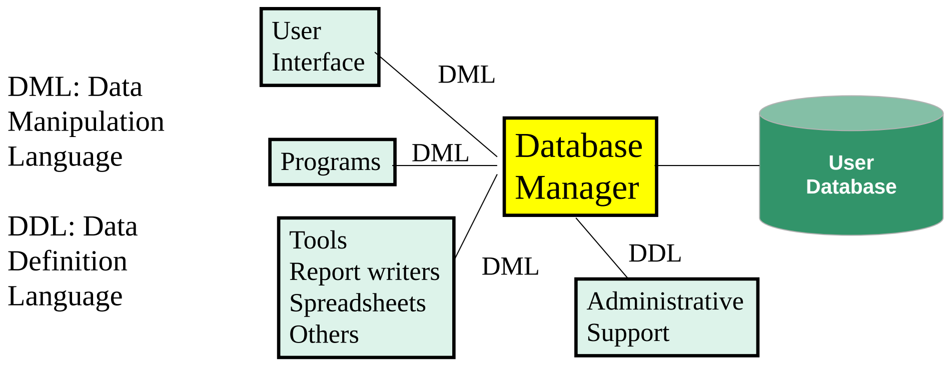
DML: Data Manipulation Language
DDL: Data Definition Language
User Interface
DML
Programs
DML
Tools Report writers Spreadsheets Others
DML
Database Manager
User Database
DDL
Administrative Support

Figure 8-4: Database Conceptual View

8.3.3 Data Models and Categories of DBMS.

A data model is a conceptual representation of data that does not include many of the details of how the data is physically stored. Database management systems have traditionally supported the following data models (see Figure 8-5):

- Hierarchical: the data model supports one to many relationships (i.e. each record has only one parent). The DBMSs which are based on this data model are called hierarchical DBMSs. IBM's IMS is an example of a hierarchical DBMS.
- Network: Many to many relationships among logical data records are supported. Example of a network database management system is Cullinets's IDMS (currently supported by Computer Associates).
- Relational: the data is viewed as tables (relations). Examples of relational DBMS are IBM's DB2, Oracle's RDBMS, and Microsoft's Access.

Two other data models gained importance since the early 1980s -- Entity-Relationship-Attribute (ERA) and Semantic data models. In the ERA data model, the data is viewed in terms of entities (objects), attributes of entities and relationships between the entities. This model is used to build a conceptual view of data in an organization (called logical data model). Semantic data model is an extension of the ERA model. The main difference is that the relationships carry meanings and the objects can inherit properties from other objects. "Object-oriented databases", discussed in section 8.7, use the semantic data model for storage and retrieval of objects and rules for complex engineering, business and expert systems applications. Other data models such as dimensional data models are used in specialized application areas such as data warehouses (see the chapter on data warehouses and data mining for a discussion of dimensional models)

Although database management systems can be categorized in a wide variety of ways, the categorization by data models is the most common. The network and hierarchical DBMS, not discussed in this tutorial, are older systems (these DBMSs flourished in the 70s). We will concentrate more on the relational and object-oriented DBMSs.

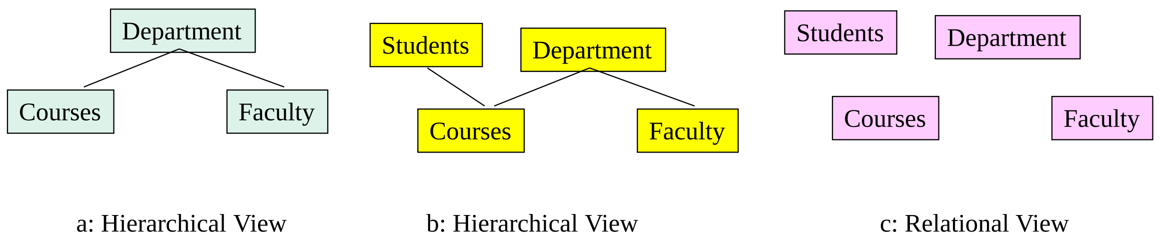

Figure 8-5: Common Data Models

8.3.4 Data View Support

A DBMS allows different users to view the same data differently. For example, information about an employee can be viewed differently by different users. The term *schema* is used in database literature to represent a data view. In a DBMS, schema exist at three levels: internal, conceptual and external (see Figure 8-6). The *internal schema* shows the physical format (e.g., text file) in which the data are stored on a storage medium. The *conceptual schema*: shows the logical layout and the relationships between data records of a database (it is also referred to as the logical data model). Database management systems have traditionally supported the conceptual schemas based on the hierarchical, network and relational data models. An *external schema*, also called a *.subschema*:, shows a user view of data. In most cases, the external schema is a subset of the conceptual schema. Figure 8-6 shows how an Electronic Health Record (EHR) database supports different views where the doctor and the administrator see only the information that is relevant to them. Support of different views of the same data is a very powerful feature of database management systems. .

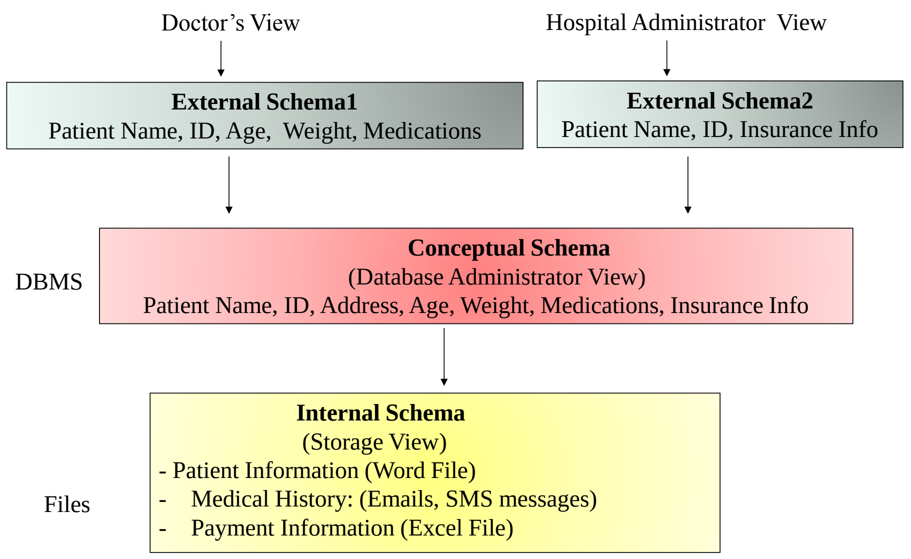
Doctor's View
Hospital Administrator View
External Schema1
Patient Name, ID, Age, Weight, Medications
External Schema2
Patient Name, ID, Insurance Info
DBMS
Conceptual Schema
(Database Administrator View)
Patient Name, ID, Address, Age, Weight, Medications, Insurance Info
Files
Internal Schema
(Storage View)
- Patient Information (Word File)
- Medical History: (Emails, SMS messages)
- Payment Information (Excel File)

Figure 8-6: Schemas (Views) in a Database for Electronic Health Record (EHR)

8.3.5 Data Definition Facilities

Data definition facilities allow creation of databases. A *Data Definition Language (DDL)* is used to define the data formats and data relationships. DDL allows data definitions at the conceptual as well as external schema levels. In addition, security and authority information is defined. DDLs can be interactive commands or batch programs. Results of DDL commands are stored in the database dictionary. In many corporations, a database administrator controls the data definition facilities for corporate-wide views, access rights, and enforcement of standards.

8.3.6 Data Manipulation Facilities

A *Data Manipulation Language (DML)* allows a user to access, manipulate and modify the database. The power and capability of DMLs depends on the underlying DBMS. Most of the modern DMLs, such as SQL, support ad hoc queries which select information and display answers on demand. DML statements may be embedded in programs written programming languages such as C, C++, Java, C# and Cobol. A DML may support report generators which produce reports with headings on special forms with appropriate printer controls. Some specialized packages such as spreadsheets, simulation packages and expert system shells may provide interfaces with database DMLs.

8.3.7 Operational Facilities

The operational facilities of a DBMS provide security, integrity and backup/recovery of a database. This includes authentication, audit trails, data consistency (the data must correctly reflect the state of a system even after failures), and concurrency (the data must be simultaneously accessible by different users). Operational facilities of a large, centralized DBMS must be comprehensive enough to allow simultaneous access of hundreds of users to large centralized databases. On the other hand, extensive operational facilities may not be needed for single user microcomputer databases. Different commercial DBMSs provide different levels of data views, data definitions, data manipulation, and operational support. An area of active work is support of databases over a network where a database can be accessed by a wide range of programs and users across a network (see Figure 8-7).

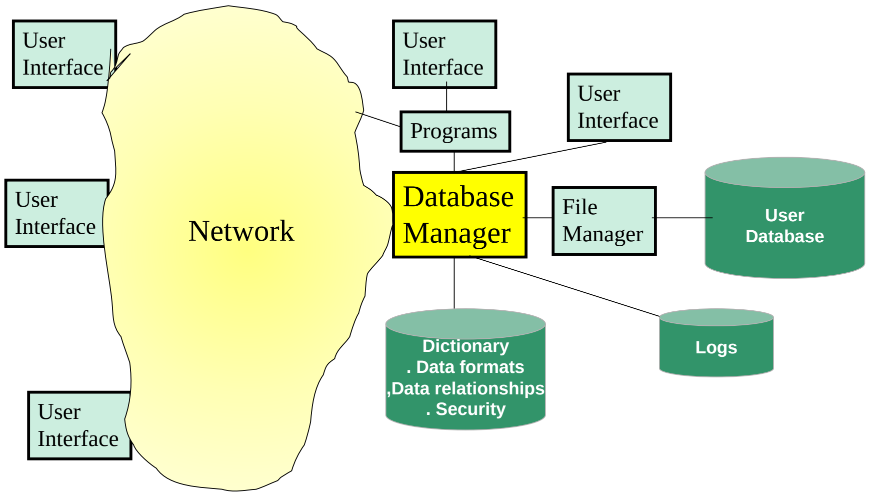

Figure 8-7: Database Management Over a Network

8.4 Overview of Relational Databases

The relational database technology, introduced by E.F. Codd [Codd 1970] at IBM, views all data as tables; all database operations are on tables; and all outputs produced are also tables. A relational database is a collection of tables. Figure 8-8 shows a relational database which consists of two tables: EMPLOYEE and OFFICE. The following terms are used in relational DBMSs (RDBMSs):

A relation is a table in which each row is unique. In addition, a relation must have a fixed number of columns. A table which satisfies these two properties is known to be in the "First Normal Form". The EMPLOYEE and OFFICE tables represent two relations in First Normal Form.

- A tuple of a relation is synonymous to a row in a table. For example, the EMPLOYEE relation has 4 tuples.
- An attribute of a relation is a table column. For example, the OFFICE relation has 3 attributes: Location, Manager and Phone.
- The degree of a relation represents the number of attributes in a table. For example, the degree of OFFICE relation is 3.

- The domain of an attribute represents the range of values for an attribute. For example, the domain for the age attribute is 0 to, say, 120 years.

Employee Table

Name	Age	Salary (K)	Location
Joe	35	42	NY
Pat	29	60	LA
Bruce	25	42	Chicago
Sam	40	75	NY

Office Table

Manag er	Phone	Location
Donna	555-1000	NY
Roger	555-1111	LA
Dave	555-2222	Chicago

Figure 8-8: Relational Tables Employee and Office

The relational DBMSs allow a user to access information from the database with only three basic operations:

- Selection
- Projection
- Join

Selection chooses rows of a table based on a criteria. For example, selection on EMPLOYEE for Age > 30 produces the rows for Joe and Sam. Projection chooses columns of a table based on a criteria. For example, projection on EMPLOYEE for the Name column lists the names: Joe, Pat, Bruce, and Sam. Join combines two different tables on a common attribute. For example, join of the EMPLOYEE and OFFICE tables on LOCATION is shown in Figure 8-9 . Theoretically, a join between two relations r1 and r2 on the joining condition r1.a1 = r2.a2 involves the following steps (a1 and a2 represent two attributes):

- Form product of r1 and r2 to produce r3'. In a product (cartesian), every tuple of r1 is concatenated with every tuple of r2 so that r3' has m x n tuples if r1 has m tuples and r2 has n tuples.
- Perform a selection on r3' where the joining attributes a1 and a2 are the same. This produces r3'', known as :hp1.equijoin:ehp1..
- Eliminate duplicate attributes from r3'' with a projection. This produces r3, known as :hp1.natural join, :ehp1. or just a join. r3 is the normal result of a join.

Joins are implemented differently by different DBMSs for efficiency. In addition to the natural joins, other forms of joins are supported in relational DBMSs. Examples are the theta and outer joins. In theta joins, also known as non-equijoins, the joining condition is r1.a1 <> r2.a2. The outer joins retrieve rows that may not meet the join conditions. This allows retrieval of data that may be lost (e.g., if joining columns have null values).

Figure 8-9: Join of Employee and Office Tables

Name	Age	Location	Manager	Phone	Location
Joe	35	NY	Donna	555-1000	NY
Sam	40	75	Donna	555-1000	NY

In addition to the basic operations of selection, projection and join, relational DBMSs allow unions, differences and intersections. A union concatenates the tuples from r1 with r2 and produce r3. Duplicate relations are eliminated from r3 as a result of union. In addition, r1 and r2 must have same number of attributes and attributes must be from same domain (union compatible). After a difference between r1 and r2, the result r3 has tuples which occur in r1 and not in r2. After an intersection between r1 and r2, the result r3 has tuples that are common in r1 and r2. Unions, intersection, differences, and products can be performed with selection, projection, and join.

In addition to these operations, some manipulation operations are introduced specifically for distributed systems. For example, the semi-join is introduced to minimize the internode traffic while performing a join of two remotely located tables [Bernstein 1981].

Relational DBMS's provide a number of attractive features:

- The relational model is simple and easy to understand.
- Desired data can be reached through a series of joins. If two tables do not have a joining column, then an "index table" can be created to facilitate joins.
- A standard query language, SQL, is used by all relational DBMS.
- Many commercial relational DBMSs are currently available for mainframes, minicomputers and workstations. Due to the popularity of relational DBMSs and SQL, many tools in business and engineering are developing interfaces to access the relational tables. This is leading to a corporate-wide database concept illustrated in Fig. E.2.
- The relational model supports "data independence" (i.e. the queries are non-procedural and do not have to know the physical data organization such as indexes and pointers).
- Relational database searches are based on data values and not on the position of data in the database. This makes data access easier.
- Relational databases and SQL are used in almost all of the currently available distributed database management systems (DDBMS).

However, relational DBMSs have a number of limitations:

- Relationships between tables cannot be modeled directly; each relationship is implicitly modeled by the inclusion of "foreign keys" as attributes. Simply stated, a foreign key enables a join between two relations. For example, the Employee-ID in the OFFICE table in Fig. E.6 is a foreign key.
- It is very difficult to represent complex design information in relational database model because relational tables do not lend themselves easily to complex data relationships, design versions and views.
- The user may be responsible for the semantic integrity of a query and completeness of an update. "Referential integrity", which ensures that all tables are modified correctly when a tuple is inserted or deleted, is not implemented in all relational DBMSs.
- The performance of queries depends on an "optimizer" which knows the internal structure of a database. It is difficult to know how well an optimizer is doing its job or if it is doing it at all.

8.5 SQL – A Quick Overview

Structured Query Language (SQL) is the standard query language for relational databases. SQL, initially also referred to as "SEQUEL", was developed at the IBM San Jose Research Laboratories in 1974. It provides interactive ad hoc queries as well as program interfaces in C, C++, C#, Java, Cobol, Fortran, ADA and PL1. The SQL language consists of a set of facilities for defining, manipulating, and controlling data in a relational database. Basically, SQL is at the core of relational da.tabase management systems (see Figure 8-10).

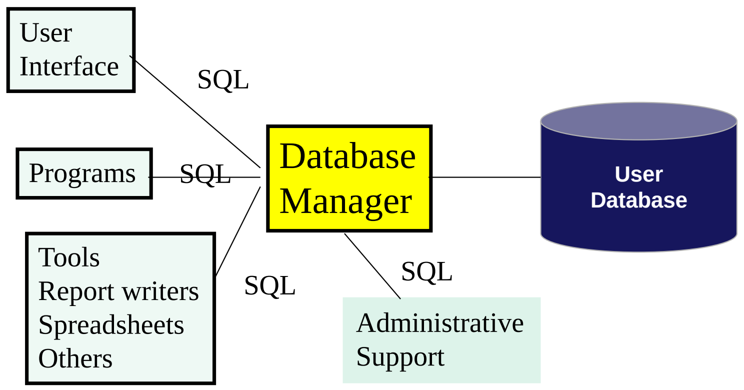

Figure 8-10: SQL is at the Core of Relational Databases

8.5.1 Data Definition

SQL data definition language (DDL) is used to create tables by using a CREATE TABLE command. The following two SQL statements are used to create a parts and a customers table:

```
CREATE TABLE parts (part_no char (4), part_name char(5)), price numeric(5))
```

CREATE TABLE customers (cust_name char(30) not null, address char(30) not null, cust_id char(12) unique not null, part_no char(4))

8.5.2 Data Retrieval

The main power of SQL lies in its data manipulation facilities. There are four basic SQL operations: SELECT, UPDATE, INSERT, and DELETE. All data retrievals are invoked by a SELECT command, which has the following general syntax:

```
SELECT   <a1,a2,a3,...,an>   FROM   <t1,t2,...,tm>   WHERE <conditions>;
```

where a1, a2,,, an are the attributes; t1, t2,, tm, are the tables; and the conditions, if specified, indicate the retrieval criteria. Conditions are specified by the "attribute op value" pairs, which can be combined through logical operators such as AND, OR, NOT. An op indicates predicates such as =, <, >, <=, >=, and <>. Examples of conditions are "age > 30", "age < 30 AND salary > 50K", etc. An SQL statement is terminated either by a ";" or by another SQL command. The SQL statements can be coded in upper or lower case. We will use uppercase letters to indicate the keywords in SQL.

The selection, projection and join operations of relational databases are performed by the SELECT statement. For example, the following statement performs relational selection (i.e., shows all columns):

```
SELECT * FROM t1 WHERE attribute op value
```

For example, " SELECT * FROM parts WHERE price > 100; " would display the rows of the parts table for prices more than 100. The statement, " SELECT * FROM parts; " would display the entire table. The projection and selection can be combined by using the following statement:

```
SELECT a1, a2,...,an FROM t1 WHERE attribute op value
```

For example, " SELECT part_no, part_name FROM parts WHERE price > 100 " would display the part_no and part_name from the parts table for prices more than 100. The joins are also performed by the select statement. The following statement causes an equijoin, where the joining condition is equality and the result include duplicates:

```
SELECT t1.*, t2.* FROM t1, t2 WHERE t1.a1 = t2.a2 op value;
```

The following statement invokes a natural join, an equijoin, which eliminates duplicates:

```
SELECT a1, a2, a3,...,an  FROM t1, t2 WHERE t1.a1 = t2.a2 op value;
```

For example, " SELECT part_no, part-price, cust_name FROM parts, customers WHERE customer.part_no = parts_no " lists the customer names who have ordered certain parts. Additional conditions can be included in joins. For example, " SELECT part-price, cust_name FROM parts, customers WHERE part_no.customer = part_no.parts and part-price >200 " would list the names of the customers who have ordered parts which cost more than $200. A product between two tables is formed by ignoring the joining condition. For example, the following statement forms a product between the parts and customers tables: " SELECT * FROM parts, customers ". Theta joins are performed by the following statement:

```
SELECT a1, a2,...,an FROM t1, t2 WHERE a1.t1 <> a2.t1 op value
```

More than two tables can be joined in a single statement:

```
SELECT a1, a2,...,an FROM t1, t2,...,tm WHERE condition1 AND condition2 AND condition3;
```

A table can be joined with itself. Aliases can be used to avoid confusion. For example, the following statement produces a list of salesmen in the same city: " SELECT first.name, second.name FROM salesperson first = salesperson second WHERE first.city = second.city AND first.ss# <> second.ss# ". In this statement, first and second are assigned as aliases. You can build complex queries by nesting queries within other queries by using the following format:

```
SELECT a1, a2,...,an FROM t1 WHERE an IN (SELECT a5,a6,...,am FROM t2 WHERE am op value)
```

For example, the statement "SELECT part_no, part_name FROM parts WHERE part_no in (SELECT cust_name FROM customers WHERE city='Detroit')" would display part numbers and names ordered by the customers who live in Detroit. The innermost query is executed first; the outer query operates on the results of the inner query.

SQL provides a powerful set of built-in functions such as ORDER, AVG, SUM, COUNT, and GROUP BY. The statement " SELECT part_no, part_name FROM parts ORDER BY part_no; " lists the part_no and part_name, sorted by part_no. The statement " SELECT AVG (price), MIN(price), MAX(price), SUM (price), COUNT DISTINCT, COUNT (*) FROM parts; " lists the average, minimum, maximum, and sum of prices. This statement will also list the distinct and total count of records in the parts table. The statement " SELECT AVG(salary) FROM employees GROUP BY title; " will produce the following display:

```
Title    Avg(salary)    secretary  1300    programmer 2500    manager   3400
```

The MINUS produces a difference (this operator is supported by some DBMSs). For example, " SELECT * FROM parts MINUS (SELECT * FROM parts WHERE price <1000) " produces the list of parts with price greater than or equal to 1000.

The predicates, used in the where clause of the select statement, provide many options, such as the following:

Comparison: =, <, >, <=, >= , <>

BETWEEN/NOT BETWEEN: An example is "where price between 5 and 20;" * IN/NOT IN: example is "where price not in (5, 7, 10);"

LIKE/NOT LIKE: these are used for pattern recognition. A "_" is used for single characters, and "%" is used for 0 to n char length. For example,"select cust_name from customers where cust_name like 'B%';" displays the names of all customers whose name starts with a B. Like/not like predicates can be used with character or graphic data.

NULL: An example is "where part_no is null;"

8.5.3 Data Modification

SQL data modification statements allow insertion, deletion and update of data in tables through the INSERT, DELETE and UPDATE statements. Here are some (hopefully) self explanatory examples:

```
INSERT INTO parts(part_no, part_name, price) (xy22, rods, 100);

INSERT INTO parts(part_no, part_name, price) (xy22, rods, null);

INSERT INTO parts-high(part_no, part_name, price) (select part_no, part_name, price FROM parts
WHERE price >1000);

DELETE FROM parts where part_no = xy20;

UPDATE PARTS set price=120 where part_no=xy22;
```

8.5.4 View Support

Views may be used to operate on portions of tables. For example, " CREATE VIEW salesperson AS SELECT name, number FROM employees WHERE job='salesperson'; " creates a portion of the employees table populated with salesmen. The " DROP VIEW salesperson; " deletes the view. One view can be created to contain columns from several tables. Views are treated as tables in SQL and can be used in any of the SQL statements. For example, views can be created from joins and can be joined with tables or with other views. Views are created temporarily; the operations are performed on actual tables. Thus updates are performed on the tables from which the views are created. Views can be used to restrict user access and handle subqueries for intermediate tables. You may create a table for more permanent operations by using statements such as: " CREATE TABLE temp1 (name, part#, price) AS (SELECT name, part#, price FROM suppliers WHERE price > = 1000); ".

8.5.5 Administrative Facilities

SQL provides two data control statements for administrators:

```
GRANT access-type ON tablename TO id;

REVOKE access-type FROM id;
```

where access-type specifies: all privileges, update, select and insert. In addition, programmers can issue "commit work" command to make changes available to others. Before the commit command, only the person entering changes sees the changes. The "rollback work" command can be used to undo changes before commit. You can modify table structure (add columns, change column width) by using the following statements (these statements are not supported by ANSI SQL):

```
ALTER TABLE tablename ADD column-name datatype;
ALTER TABLE tablename MODIFY  column-name datatype new-width;
```

8.5.6 Embedded SQL

SQL statements can be embedded in host programs written in several languages such as C, C++, C#, Java, and many older languages such as Cobol, Fortran, and PL1. The SQL statements in programs are embedded by using the EXEC SQL statements in a program:

```
EXEC SQL  sql statements
```

Two unique problems are concerned with embedded SQL: connecting the SQL variables with programming language (host) variables and handling of multiple rows returned from SQL statements. Connection with host variables is established by reading the selected attributes INTO a set of host program variables:

```
EXEC SQL SELECT  a1, a2,,,, an       INTO :p1, :p2,,,,,:pn       FROM t1  WHERE condition;
```

The host variables p1, p2,,, pn are indicated by a ":". For example, the statement " EXEC SQL SELECT part_no part_name INTO :pnumber, :pname FROM parts " will store the part_number and part_name attributes from table parts into host variables pnumber and pname.

A "cursor" is used to handle multiple rows returned from SQL. The problem is that the traditional procedural languages are record oriented; they process one record at a time. However, SQL may return many rows as the result of a single embedded select statement. A cursor is first declared for an SQL statement to be executed. It is then opened in a manner similar to a file open. The program then issues FETCH statements to retrieve the rows returned. SQLCODE, a flag, is checked to see if any rows are left to be retrieved. The following code is an example:

```
EXEC SQL DECLARE cs CURSOR FOR <SQL statements>
EXEC SQL OPEN cs
EXEC SQL FETCH cs INTO <host variables>
CHECK SQLCODE for end of fetch
```

Figure 8-11 shows the sample pseudo C/C++ code which illustrates how embedded SQL can be used to extract information from the parts table defined by the shown CREATE command. The sample code has three segments. The first segment shows how the host variables are defined by using an int statement. The inclusion of SQLCA brings many of the SQL flags and variables (e.g., the SQLCODE which shows return codes) into the program code. The second segment of the code shows how the information for part number 75567 is accessed and printed. The third part of the code shows how multiple rows are processed by using the cursor statement. The reader should understand that the code shown here is given for illustrative purposes only. Differences exist between different languages and vendors.

Table definition:

```
        CREATE TABLE parts (part_no numeric (4), part_name char(5)), part-
          price numeric(5)) ;
```

Sample code to Extract and display information from parts table;

```
    Include SQLCA

    Int p-number, p-price;

    …

    Exec SQL select part-no, part-price into :p-number, p-price from parts where
       part-no =75567;

    Print (p-number, p-price);

     …..

    Exec SQL cursor CS

             select part-no, part-price into   where part-price >200;

    Exec SQL select part-no, part-price into :p-number, p-price from parts where
       part-price =200;

    Exec SQL open CS;

    Do {Exec SQL Fetch CS into :p-number, p-price; } while sqlcode =0;

            Exec SQL close CS;
```

Figure 8-11: Sample Embedded SQL Code

8.5.7 Performance

SQL query optimization is the responsibility of DBMS which parses the query and then executes it in an appropriate manner. Clever techniques which rely on internal organization are not recommended. A user can create an index for fast access. For example, if the parts table needs to be accessed by part_no frequently, then an index on this column will speed up the performance. The following statements create and drop an index:

```
CREATE INDEX indexname ON tablename (column-name);        DROP INDEX indexname ON
tablename;
```

Many indices can be created on one table. The user does not specify when and how the index will be used. SQL determines when to use an index (recall that a SELECT statement does not include any reference to an index). An index is not used if a WHERE clause is absent in queries. The use of the index to satisfy a query is decided by the query optimizer.

Joins are the major source of performance problem. For an m row table join with an n table join:, there can potentially be mxn operations. Over the years, local optimizations for joins work fine, however, distributed joins can create major performance problems. In addition, you should minimize duplicate values to avoid too many updates.

Building Simple Database Apps by Using MS-Access

MS Access is organized around databases that consist of tables. Queries are created on tables.

First start MS-access from your computer and create a folder first in which you will store the databases, the tables, and the queries.

. To create a database.

- On the main menu, choose file->new and then choose a 'blank database'
- Give a database name (say db1)
- MS-access will create a database in which you can create your own tables (one database can have many tables).

To create a table

- open a database (db1)
- MS access will show a menu of options to create tables. Choose "create table in design review"
- Choose a table name
- Define the fields (attributes) of the table by using the GUI.
- For each field, you can choose its name, its type (text, numeric), length (shows at the bottom part – you can edit it)
- when done, close (save) it

To add records to the table:

- Open the table (click on "tables") and then select the table name you want
- From the screen add records to the table
- You can add as many records as you want
- Close the table and give it a name (e.g., table1)

To create a query

- Open a query (click on "queries'')
- Choose "create a query in design view"
- Choose the table ("table1") on which you want to create queries
- Develop a query by using the GUI
- To see SQL created, get to 'SQL view' by using right click on grey area.
- Edit the query (if needed) and save it

To run a query:

- Click on a query. MS Access will run it.

Suggested Approach: MS Access is a simple database management system that is available as part of the MS Office. This short tutorial is intended to be release independent because MS Access changes with every release. For additional information, please review the large number of MS Access tutorials that are available on the Internet.

8.6 Overview of Database Design

Database design attempts to provide consistent and current information to the end users in a speedy fashion. The design process goes through the following general steps, illustrated through a simple example in Figure 8-12:

1). Capture Information Requirements. The information requirements of different users and applications are developed after interviewing different sets of users.

2).Build a Logical Data Model (LDM). An LDM represents user data requirements and contains the following pieces of information:

- Entities (objects) such as customers, parts, products, students
- Attributes of entities such as the name and address of customer
- Relationships between objects such as customers buy products.

Different people may develop different views of LDM (user view, management view, programmer view). These views are usually integrated to create a common corporate LDM. In addition, the LDM is cleaned up to remove synonyms, homonyms, and derived data. The resultant LDM can be represented as an ERA diagram.

3). Develop a Database Design. This step first creates a "normalized" database structure. Normalization is a procedure for decomposing large data entities to remove update anomaly (should not have to update more than is needed) and delete anomaly (should not delete more than is needed). Normalization, not needed for retrieval only data, should be in 1st, 2nd and 3rd normal forms. Discussion of normalization is beyond the scope of this book. See the side bar "An Informal Normalization Example" for the key ideas.

After normalization, known as logical database design, a physical database design translates the normalized logical data model into a DBMS supported physical structure. The physical design depends on the DBMS type. In case of a relational database system, the physical design shows the tables, the attributes of each table, and the indexes (see section C3). Detailed discussion of this topic is also beyond the scope of this book.

In a distributed environment, the design also includes the following steps.

4). Data Partitioning and Clustering. This step attempts to decompose data for application/user specific requirements. For example, as shown in Figure 8-12, the customer table is partitioned into two regions because the customers live in two regions. The partitions may also be clustered because some information is always used together.

5). Data Allocation. The clustered data is allocated to different physical sites to minimize response time and improve availability. A very large number of data allocation algorithms have been developed in the academic community under the heading of "file allocation problem (FAP)".

The book "Database Design" by Toby Teorey (Prentice Hall) is an excellent source of information on this topic.

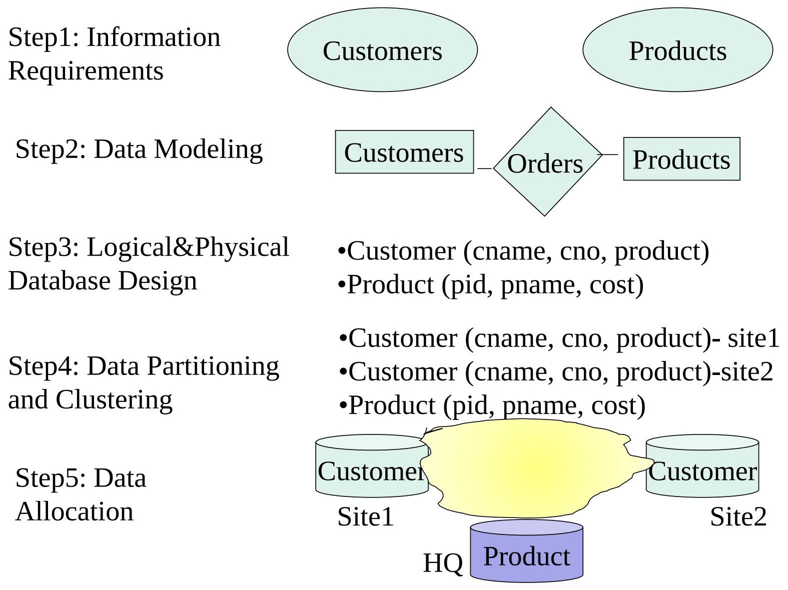
Step1: Information Requirements
Customers
Products
Step2: Data Modeling
Customers
Orders
Products
Step3: Logical&Physical Database Design
•Customer (cname, cno, product)
•Product (pid, pname, cost)
Step4: Data Partitioning and Clustering
•Customer (cname, cno, product)- site1
•Customer (cname, cno, product)-site2
•Product (pid, pname, cost)
Step5: Data Allocation
Customer
Site1
Customer
Site2
HQ
Product

Figure 8-12: A Database Life Cycle

8.7 Object-Oriented and NoSQL Databases

8.7.1 Introduction

Relational databases are suitable for many applications. However, it is not easy to represent Big Data and complex information in terms of relational tables. For example, a car design, a computing network layout, and software design of an airline reservation system cannot be represented easily in terms of tables. For these cases, we need to represent complex interrelationships between data elements, retrieve several versions of design, represent the semantics (meaning) of relationships, and utilize the concepts of similarities to reduced redundancies. Over the years. the O*bject-Oriented Database Management Systems (OODBMSs)*, have been developed to support applications in computer aided design and computer-aided manufacturing (CAD/CAM), expert systems, computer-aided software engineering (CASE), and office automation. Simply stated, OODBMSs combine and extend the features of database management systems, artificial intelligence, and "object oriented programming" for these and other future applications.

In addition to the OODBMS, we have seen the rise of a new type of databases, known as *NoSQL databases*, that are also challenging the dominance of relational databases. NoSQL means Not Only SQL, implying that SQL is not the only solution. Historically, NoSQL was a hashtag (#nosql) chosen for a meetup to discuss these new databases. There is no formal definition of NoSQL but it represents the following common situations:

- Not restricted to the relational model and SQL
- Should include complex objects handled by OODBMS but not be restricted to OO view

- Should be able to handle Big Data sources that include structured, semi-structured and unstructured files
- Should be able to run on clusters of computers and support Hadoop type frameworks
- Should be open-source and designed for the latest web accesses hema-less

There is some debate about OODBMS versus NoSQL databases but in fact NoSQL includes OODBMS. Many database vendors, such as RedisLabs, are offering both. We will first focus on OO databases because they are more developed and will then briefly review NoSQL databases.

8.7.2 Object-Oriented Databases[1]

Object-oriented databases allow storage and retrieval of non traditional data types such as bitmaps, icons, text, polygons, sets, arrays and lists. The objects can be simple or complex, can be related to each other through complex relationships, and can inherit properties from other objects. Object-oriented database management systems (OODBMS), which can store, retrieve and manipulate objects, have been an area of active research and exploration since the mid 1980s. Most of the work in OODBMSs has been driven by the computer aided design and computer aided manufacturing (CAD/CAM) applications.

OODBMS allow complex relationships between data entities. They combine several features of DBMS, AI and software engineering. In an OODBMS, the data is viewed as objects with the following stipulations:

- Simple object is a relational table
- Complex objects are built from simple objects
- Each object has properties (attributes)
- Objects are inter-related through complex relationships
- Relationships may carry semantics (meaning). This primarily involves two types of relationships: inheritance (is-a and a-kind-of –ako) and aggregation (part-of and contains)
- DDL may use inheritance to create new objects
- The DML can be customized with a general query language that uses AI pattern matching
- You can store procedures with the data

[1] This section requires some understanding of object oriented concepts.

Object-Oriented Database Manifesto
OODatabase Conference (Kyoto, Japan,1989)

Object-oriented Features	**Database Features**
• Complex objects	• Persistence
• Object identity	• Secondary storage management
• Encapsulation	• Concurrency
• Types and classes	• Recovery
• Inheritance	• Ad hoc queries
• Overriding, overloading, and late binding	
• Computational completeness	
• Extensibility	

Figure 8-13: The Object Oriented Database Manifesto

What exactly is an OODBMS? This question has been asked since the mid 1980s. In 1989, a group of computer scientists got together and established "The Object-Oriented Database Manifesto" [Atkinson 1989]. This Manifesto, displayed in Figure 8-13, establishes the basic properties of OODBMS by combining conventional database functionalities with object-oriented functionalities. According to this manifesto, the key properties of OODBMSs are:

Data may be stored, retrieved and manipulated as complex objects which consist of sets, lists, arrays or relational tuples.

- OODBMSs allow creation of objects from existing objects by using inheritance of properties.
- Procedures can be stored as objects in the database.
- The relationships between objects can be complex, many to many relationships.
- Objects can contain very large values to store pictures, voice or text.
- Most OODBMSs provide facilities to track multiple versions of an object.
- In many OODBMS, the data manipulation language is closely related to the programming language

The OODBMS systems generally fall into two categories:

- Extensions of the object-oriented programming languages (OOPL) to include the features of DBMS. Examples are O2 and Gemstone systems.
- Extensions of the relational DBMS to include the features of OOPL. Examples are Starburst and POSTGRESS.

OODBMS have moved from state of the art to state of the market, however they are not heavily used at the time of this writing (less than 5% of corporate data is stored in OODBMS). Objectstore is one of the most popular OODSBMS. It is beyond the scope of this book to discuss detailed features of existing OODBMS.

Advantages/Disadvantages of OODBMSs. OODBMSs have emerged due to the limitations of relational DBMSs in handling complex relationships and semantics. In addition, OODBMS attempt to include desirable features from AI and software engineering to improve application reusability and maintainability. Despite several potential advantages of OODBMSs, a few concerns should be noted. First, standard query languages for OODBMS such as OQL are not very popular commercially. In addition, the performance characteristics of OODBMSs are not well understood. This problem is expected to be addressed by performance improvements in the OODBMSs. We have to see how OODBMSs operate in large applications with thousands of users.

Objectizing a RDBMS. OODBMSs have not been as successful as relational databases so far. However, OO views on relational databases is a very successful approach. The main motivation is that most programs work on objects and should not have to translate between object and relational views. Basically, RDBMS operations need to be performed on objects (i.e., fields), on sets of objects (i.e., rows), and on tables (i.e., joins). The different approaches to "objectizing RDBMSs" are:

- BLOB (binary large objects) support in RDBMS. Blobs allow you to store, retrieve, and display graphics, memos, and video clips. However, BLOBS do not allow querying the content of the BLOB (i.e., you can store and display the pictures of employees but cannot select the employees who have images).
- View relational data as a collection of methods provided by the RDBMS. For example, the SQL statements such as create, select, update, insert, delete, and grant can be thought of as methods that are performed on the data.
- Build wrappers around SQL (i.e., treat each SQL statement as an OO statement). The wrapper contains classes to create a table, to insert a record, to read arecord, to examine the status of an object, and to do do several other low level database actions. Warappers of this type are becoming commercially available (e.g., DBTools.h++ from Roguewave). This is not a problem in static SQL because in this case, the variablesa are bound at startup time and type checking is done at compile time. However, for dynamic SQL, this is tough to do because how can you know about variables and type checking.
- Build wrappers around the database itself (i.e., make tables look like a set of objects and perform operations on them). These wrappers are more sophisticated and map application objects to relational tables (some wrappers provide the reverse functionality also). These wrappers support encapsulation (i.e., methods to define the inetrfaces to the databases, inheritance (i.e., the sharing of attributes, queries, methods, and relationships between objects); and associations (i.e., the relational foreign key relationships are associated to relationships among classes). These wrappers may also map relational database capabilities to object classes. And support object caching, transactions, and database efficiency. The wrappers provided by Persistence Software fall into this category.

8.7.3 NoSQL Databases – A Quick Overview

As stated previously, NoSQL means Not Only SQL, implying that SQL is not the only solution for database access. NoSQL developments are being heavily fueled by the popularity of Big Data applications. While SQL databases work well for ERP applications, they are not suited for handling large volumes of unstructured data such as emails, social media, video clips and satellite feeds. At present, NoSQL databases can broadly be categorized in four types.

- Key-Value Databases: These are the simplest NoSQL data stores that allow the clients to get the value for a given key, insert a value for a key, or delete a key from the data store.

- Document Databases. These databases allow the clients to store, search and retrieve documents, which may consist of unstructured (e.g., Word) or semi-structured (e.g., XML, JSON) documents.
- Columnar Databases. These databases allow storage of commonly used data in column families as rows that can be accessed quickly. For example, a student overall profile (name, ID, area of study, GPA) is needed more often than the detailed transcripts. These databases are generally useful for content management systems.
- Graph Databases. These databases allow the clients to store entities (nodes) and relationships (edges) between these entities. Although graphs can be stored in RDBMS, graph databases are designed to traverse through the joins or relationships very quickly. They can handle semantic analytics also because semantic information about relationships can be easily added.
- Object Oriented Databases. These databases, discussed previously, can be used to represent complex design information such as car or network design.

Many NoSQL products are being announced at the time of this writing. Detailed technical analysis of NoSQL databases is beyond the scope of this chapter. A large number of video clips, blogs and articles on NoSQL can be found on the Internet for better insights.

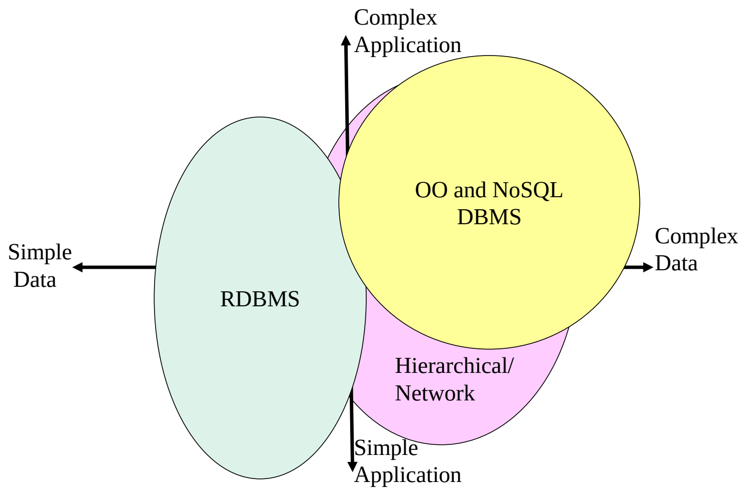

Figure 8-14: A Model for Evaluating Database Technologies

Basically, NoSQL databases are specialized for specific purposes instead of the general nature of relational databases. The popularity of NoSQL databases does not mean the demise of RDBMS. The main question is which database technologies are suitable for which applications? Figure 8-14 attempts to answer this question by using the data and process complexity of applications as a measure. For example, the x-axis shows the complexity of the data model (one to one versus many to many relationships) and the y-axis shows the complexity of the processes (simple retrieval and storage versus complex computations). This figure shows the regions where some of the database technologies can be most effective. For example, it shows that the older network and hierarchical DBMSs are more suited for complex data but relatively simple processing type applications while the relational DBMSs are more suitable for applications with relatively simple data models. Theoretically, OODBMSs are intended for the complex data and complex processing type applications. However, Big Data applications include all of the above. In other words, to meet the challenge of Big Data, all types of database technologies will be needed and newer technologies will be introduced.

Data Quality — a Major Issue

Data quality has emerged as a major issue recently due to its potentially severe impact on effectiveness of an organization. For example, a leading computer industry information service indicated that it expects most business process reengineering initiatives to fail through lack of data quality [Wand 1996]. In addition, wrong price data in retail databases may cost American industry as much as $2.5 billion in overcharges annually [English 1996]. However, state of the practice in data architectures typically does not include data quality issues. The data architecture work should explicitly include the following two data quality considerations:

- How to measure data quality (i.e., metrics), and
- How to improve data quality.

There are different views and definitions of data quality (see, for example, [English 1996, Wand 1996, Moriaritty 1996]). We will use the following operational definition of data quality [Redman 1992]:

"A product, service, or datum X is of higher quality than product, service, or datum Y if X meets customer needs better than Y."

This definition needs further refinement and elaboration. In particular, we need to focus on the following quality attributes of data:

- Accuracy: reflects correctness to real life (e.g., a customer address reflects where he/she lives).
- Consistency: two or more things do not conflict with each other (e.g., two different addresses for the same customer).
- Currency: recency of information (e.g., the customer address is the current and not a previous address).
- Completeness: degree to which values are present in a data collection (e.g., the address shows the street, city, and zip code).

These attributes provide the core data quality metrics. Measurement of these metrics and improvements in data quality by using these metrics as a yardstick is the main challenge of data quality work at present. The data architects should use these metrics to improve the quality of data with concomitant improvement in business processes that rely on data.

The approaches to obtain data quality fall into the following broad categories:

- Data cleanup
- Process cleanup

Data cleanup involves use of a tool to identify "bad data" (i.e., not accurate/consistent/current/complete) and then, the elimination of bad data through automated and/or manual processes. A wide variety of "data scrubbers" are commercially available from vendors such as Vality to perform this task. However, data cleanup needs to be a periodic effort that must be repeated several times over the life cycle of data. It is better to establish a process that cleans the data and keeps it clean. This process must be built into the data architecture steps.

Time to Take a Break

Suggested Review Questions Before Proceeding

- What are the basic data concepts
- What is enterprise (corporate) data and
- What is Big Data and how is it related to enterprise data in modern organizations
- What are Database Management System (DBMS)
- What are relational databases and what is SQL
- What are the basic principles of database design

8.8 Data Warehouses for Business Intelligence

8.8.1 Overview

Very large databases and systems require special capabilities and tools to analyze large quantities of data and for business intelligence. As stated previously, a data warehouse is a repository of information (data) for decision support and business intelligence. The notion of data warehouse was first introduced by Barry Devlin and Paul Murphy [Devlin 1986] with statements such as the following:

> "To ease access to the data..., it is vital that all the data reside in a single logical repository, the Business Data Warehouse".

This concept was commercialized by IBM as "Information Warehouse" in 1991 (see IBM Programming Announcement, September 11, 1991; and "Information Warehouse: An Introduction", IBM Manual GC26-4876). The IBM Information Warehouse strategy provides for direct access to operational data maintained by on-line systems and promoted the notion of "universal data access". This strategy was supported by a variety of protocols and data access mechanisms to be supplied by a multitude of vendors. However, the complexity and inadequacy of accessing non-relational data such as IMS data from SQL-based tools deterred many implementations of Information Warehouse.

The term "Data Warehouse" was popularized by Bill Inmon [Inmon 1993] to emphasize the separation of ***operational data*** (the data that is used for daily transaction processing) from ***informational data*** (data used primarily by decision support users such as executives, analysts, and line managers). In many ways, a data warehouse is being used as a modern term for a collection of reporting systems. Data warehouses are developed primarily for decision support systems (DSS) rather than the traditional on-line-transaction processing (OLTP) systems.

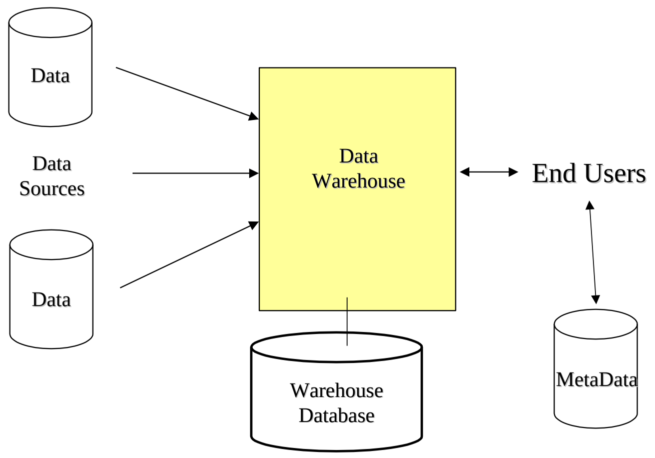

Figure 8-15: Conceptual View of a Data Warehouse

Figure 8-15 shows a conceptual view of a data warehouse. The basic components of a data warehouse are:

Warehouse database that contains information to support corporate decisions. This logical database is the center piece of a data warehouse and is typically partitioned and organized by subjects (subject-oriented databases to reflect customers, prices, products, etc.). The warehouse database may be distributed to several sites (most corporations at present have one centralized data warehouse database). This database is primarily used for retrieval and typically contains:

- Summary information (lightly summarized or highly summarized)
- Detailed data
- Historical data

Data sources that are used to populate the data warehouse. These databases typically consist of operational databases but may also include many other data sources such as external databases (databases that are external to the enterprise) and unstructured data (e.g., memos and meeting minutes).

Extract program that is used to load the data warehouse. The extract program may populate the data warehouse periodically (e.g., every midnight) or due to an event trigger (e.g., every time a customer's payment is received, the extract program sends this information to the data warehouse).

Meta data (data about data) defines the informational data contained in the warehouse in user and/or business terms. Meta data helps the users to issue business related queries without having to know the physical intricacies of the data. Meta data presents a comprehensive data dictionary and other data description and access information, based on a single, integrated data model.

Users of data warehouses who employ a variety of decision support tools to access the data warehouse. Decision support tools typically include:

- Report writers
- Data browsers

- Spreadsheets
- 4GLs (Fourth Generation Languages)
- "Drill down" applications
- Other planning and modeling tools

8.8.2 Definitions

Data warehouse is a repository of information for decision support.

Information warehouse is the same as data warehouse.

Data mart is a regional data warehouse that may serve a region or a department of an organization.

Data sources are used as inputs to create a data warehouse. Data sources may comprise of structured data (data that is stored in databases) or unstructured data (data that is contained in memos, diagrams, market reports, etc.).

Operational data is the data used to support the day-to-day operations of a company.

Informational data is the data used to support business decisions.

8.8.3 Examples of Data Warehouses

The following examples from different industry sectors illustrate some sample data warehouses.

1) A large bank pays $2 per card per month to VISA for the use of VISA logo and services (i.e., $24 per year for every card holder). The bank wanted to know

> "How many VISA card holders from our bank did not use their VISA card last year".

The answer was: 500,000. This meant that the bank paid $12 million dollars to VISA for nothing (500,000 x 24 = 12 million). The bank cancelled these customer cards and saved $12 million dollars per year.

2) A healthcare organization queries quarterly insurance claims for exceptions to the norm. The typical queries used` are:

> "Which doctors in California charge more than the national average for a broken leg procedure?"
>
> "Which doctors filed the largest number of claims during the last quarter?"

A data warehouse answers these queries, resulting in improved cost monitoring.

3) A materials director at Hughes Aircraft asked the following questions:

> "How many purchase orders were placed in the last three years for the Sector, and what were the corresponding dollars?"
>
> "What are the top 10 purchased commodities and corresponding suppliers in the Sector?"

The Hughes Aircraft operational data was spread over 8 computing platforms, interconnected through 7 types of networks, stored in 11 DBMSs, and owned by 630+ applications. It was virtually impossible to answer these queries from the existing operational databases. A materials data warehouse was established with considerably improved materials management system.

4) A retail department chain store tracks and analyzes sales by querying a stores data warehouse with questions such as the following:

"Which products in my store are selling most quickly?"

"Which products stay in the inventory the longest?"

"How does one store's profitability compare to the rest of the chain?"

Answer to these queries helps the retail store in establishing corporate directions.

8.8.4 Overview of Data Warehouse Architectures

The data warehouse view presented in Figure 8-15 is conceptual. In practice, the data warehouse can be architected in a variety of ways as shown in Figure 8-16 through Figure 8-18.

8.8.4.1 Localized Functional Warehouses ("The Data Marts")

The localized data warehouses, also known as the "data marts', are typically created by individual departments or divisions to support their own decision support activities (see Figure 8-16). These data marts may be created to support specific products (e.g., automobile parts) or function (e.g., loan management) of individual departments, divisions or regions. In some cases, data marts may be created for user populations with the same technical environments. For example, separate data warehouses for PC, Apple or Mobile Apps could be created.

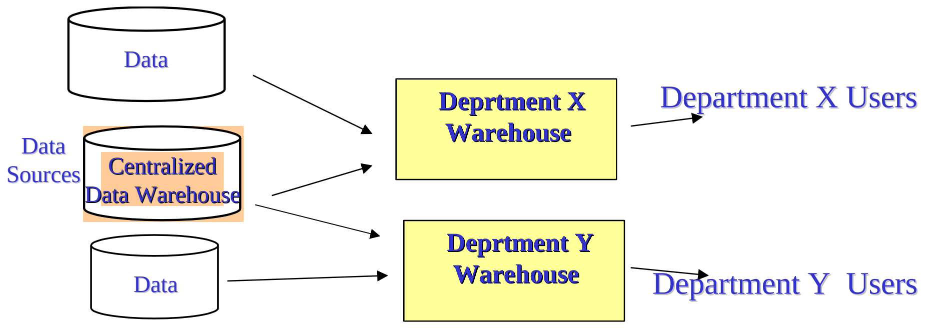

Figure 8-16: Localized Data Warehouses (The Data Marts)

The source data for these data marts usually comes from the department/region's operational databases. A small portion of the local warehouse data may come from "external" databases from other departments/divisions and companies. Once created, there is no coordination between the different data warehouses, even though the stand-alone data warehouses may submit data to corporate site for consolidation and global reporting. The data in each stand-alone data warehouse is not likely to be consistent with corporate data or with data in other stand-alone data warehouses. This is primarily due to the differences in focus, level of detail, and end-user interests.

The primary advantage of data marts is that they can be developed quickly to serve the local needs without having to wait for the large corporate data warehouse. The obvious disadvantage is the proliferation of data warehouses that are not consistent with each other. In practice, stand-alone data marts can be used by organizations with very independent and "non-intersecting" departments as a starting point in an overall strategy for a centralized corporate data warehouse.

8.8.5 Centralized Data Warehouse

The centralized data warehouse approach, shown in Figure 8-17, is the most common approach to building a data warehouse. This approach, popularized by IBM's "Information Warehouse", advocates a large centralized warehouse database that adheres to a single, consistent enterprise data model. All operational and external data is copied and stored in the central data warehouse. The central warehouse may be used to populate individual data marts for improved performance and ease of access.

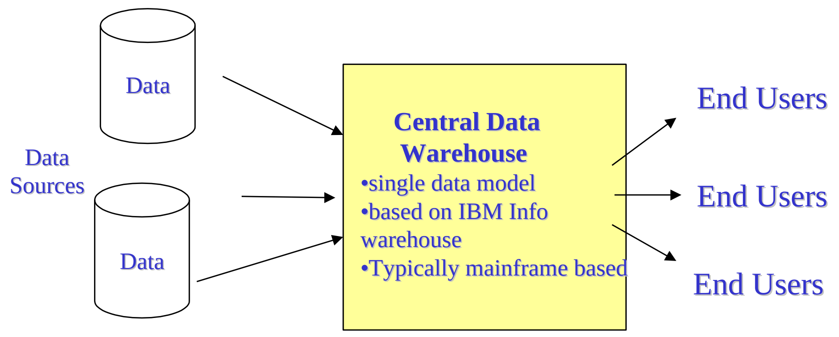

Figure 8-17: Centralized Data Warehouse

The primary advantage of this architecture is that the same consistent and complete data is utilized by all users. The users need to logon to one environment and are thus freed from having to worry about data stored in different platforms and environments (see the British Telecom case study later in this chapter). This approach also works well for enterprises where most of the processing is done at the corporate site. The main drawback of the centralized data warehouse is that it is very difficult to develop a global data model for most organizations (imagine building a global data model for General Motors!). It is also difficult to agree on a corporate wide level of detail and naming conventions. Organizations must also carefully manage the performance and end-user access to centralized warehouses to ensure that the users continue to rely on the centralized data warehouse.

8.8.6 Distributed Data Warehouses ("The Virtual Warehouse")

Many data warehouses can be distributed in an enterprise in a manner similar to the file and database servers. These distributed data warehouses (distributed data marts) can be accessed by the end-users through a "warehouse front-end" as shown in Figure 8-18. The front-end, a "data warehouse mediator", usually contains a global data dictionary that knows the location and format of the needed data and how to send the queries to the final destination. In some cases, the front-end can send the requests directly to the operational databases, thus eliminating the need for some data warehouses. This approach also allows the creation of a "virtual" data warehouse that simply routes the user queries to the following data sources:

- Operational databases for detailed information
- Archival databases for historical analysis
- Data warehouses for highly and lightly summarized data

The virtual data warehouses essentially employ the read-only capabilities of distributed data management technology. Ideally, the virtual data warehouse should have the intelligence to automatically migrate the data from the data sources to the data warehouse. You can thus start with a small version of a data warehouse that automatically grows with use as needed data is transferred to it gradually. The data in the virtual data warehouses may be uniquely assigned or may be partially duplicated to improve performance and availability. A global corporate data warehouse can play a key role in this architecture. It can contain data that is common across the corporation. It may feed the local warehouses or may also be fed by local data warehouses.

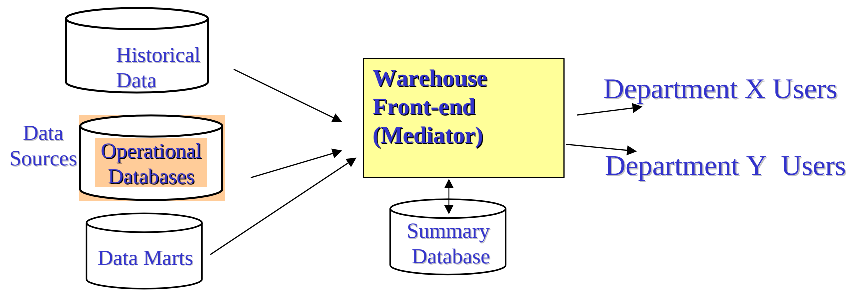

Figure 8-18: Distributed Data Warehouse ("Virtual Data Warehouse")

The distributed data warehouses provide many benefits in terms of flexibility, performance, scaling, and load balancing. The main appeal of the distributed data warehouse is that it can be designed to match the topology of most organizations, i.e., a corporate data warehouse at the corporate site and regional data warehouses (the data marts) at the branch offices. In this case, all local queries go to the regional data warehouses and all corporate queries go to the corporate data warehouse. In addition, different data marts may employ different technologies, e.g., one data mart may use a specialized DBMS suitable for one type of subject.

A potential disadvantage of this approach is that a global data model may still be needed to populate the warehouse front-end. In addition, if the databases to be accessed through the front-end are widely distributed, then significant performance degradation and service outages can occur. In general, distributed data warehouse design is complicated and requires considerable trade-offs in data distribution and query optimization.

8.8.7 Hadoop – A Big Data Warehouse

Hadoop is an open-source software framework, developed by Apache, for distributed storage and distributed processing of very large data sets such as Big Data. Hadoop is a highly distributed open-source system that runs on many computer clusters built from commodity hardware (e.g., Dell computers) instead of one large scale computer system. Since its inception in 2003, the Apache Open Source Foundation has been developing and refining Hadoop. Although many variations of Hadoop have been developed by commercial organizations such as SAS and several alternatives to Hadoop have been proposed by other developers, Hadoop has become the center of activity for Big Data applications. Basically, the term Hadoop refers to Hadoop core plus the collection of additional software packages – called the Hadoop ecosystem -- that can be installed on top of or alongside Hadoop.

The idea of Hadoop is based on the Google File System paper that was published in October 2003. This paper triggered another research paper from Google on the MapReduce algorithm. Despite many changes and extensions, the architecture of Hadoop is based on three core building blocks: Hadoop Distributed File System (HDFS), Map-Reduce Algorithm and a Scheduler. Figure 8-19 shows a conceptual view of the Hadoop Architecture in terms of these three building blocks. Basically, Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes that provide distributed storage and computation across clusters of computers. This distributed architecture is designed to process large volumes of Big Data and scales up from single server to thousands of machines, each offering local computation and storage. A simplified overview of Hadoop architecture is presented below.

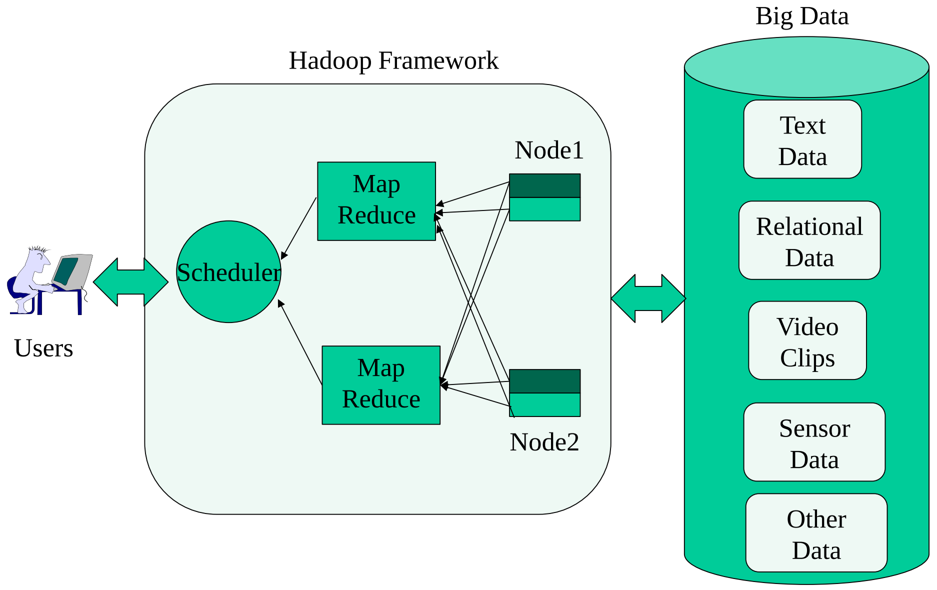

Figure 8-19: Hadoop Framework Conceptual View

The Hadoop Distributed File System (HDFS) is based on the following concepts:

- A file system based on a cluster of commodity computers working together to present high-performance, fault-tolerant disk storage
- An open-source implementation of the Google File System
- High performance for reading large files that are written once and read many times
- Extensive use of replication to ensure that files remain accessible even if one or more of the servers in the cluster go offline
- Not designed for small files or for random access within files

Hadoop MapReduce module heavily relies on clusters of commodity computers and a distributed file system such as HDFS to enable massively parallel processing of data. Specifically, MapReducejob consists of two functions:

- The Map function that first converts the tasks to maps that run locally on the data located on multiple machines (in parallel, of course)
- The Reduce function that combines the output of the Map functions and produces the final output

A Job Tracker (Scheduler) supports the MapReduce processing by accepting jobs and distributing the Map and Reduce tasks to Task Trackers on each machine. The job Tracker is a sophisticated scheduler that tries to keep the work as close to the data as possible. With the assistance of a rack-aware file system, the JobTracker knows which node contains the data, and which other machines are nearby if the work cannot be hosted on the actual node where the data resides. his reduces network traffic on the main backbone network.

Due to a highly distributed architecture and sophisticated scheduling algorithms, Hadoop can handle extremely large Big Data files with very diverse content that spans text files, relational databases, video clips and sensor data. A great deal of information about Hadoop is available on the Hadoop Official Site (*www*.hadoop.apache.org). The following sources are recommended.

- Apache Software Foundation. (2013, 08 04). HDFS Architecture Guide. Retrieved from Hadoop: http://hadoop.apache.org/docs/stable1/hdfs_design.html
- Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. Retrieved from Google Research: http://static.googleusercontent.com/media/research.google.com/en/us/archive/mapreduce-osdi04.pdf
- Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google File System. Retrieved from Google Research: http://static.googleusercontent.com/media/research.google.com/en/us/archive/gfs-sosp2003.pdf
- Rajaraman, A., Leskovec, J., & Ullman, J. D. (2013). Mining of Massive Datasets. Palo Alto, CA: Stanford University

8.9 Business Intelligence, Business Analytics and Data Mining

8.9.1 Using the Data to Make Decisions

Our interest is how data, in whatever form, supports decisions at various levels in an organization. Specifically, we want to review the variety of decision support tools that are currently available to support business intelligence (BI), analytics, and data mining operations.

Figure 8-20 shows a conceptual view of the decision support tools that use a variety of data sources (e.g., data files, databases, data warehouses and miscellaneous data sources such as satellite and social media data) to produce outputs such as reports, dashboards, charts, graphs and other visualizations. The decision support tools maybe called BI tools, analytics tools, visualization tools and the like. In other words, we will use decision support as a general umbrella term to discuss the variety of tools that can be used in a data driven organization.

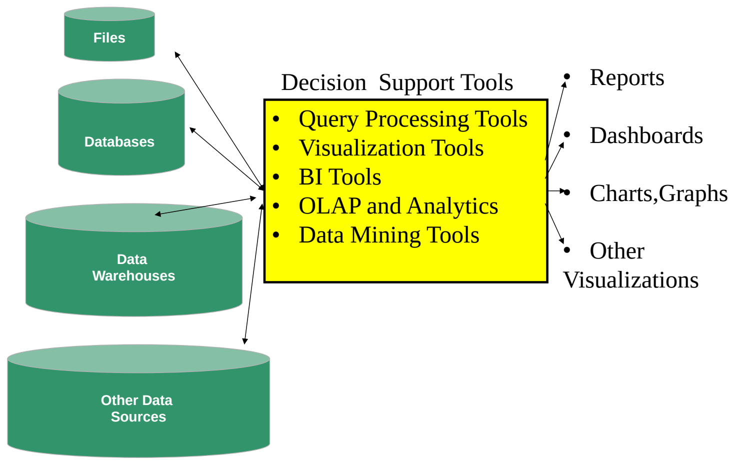

Figure 8-20: Decision Support Tools

Decision support systems (DSS) ideally operate in an exploratory manner that is driven by the desire to discover the unknown. Computerized DSS provide powerful capabilities for supporting a very wide range of decisions very quickly and improve communication and collaboration by overcoming cognitive limits of decision makers interconnected through the Internet. Although DSS can use any type of data source, most DSS are built around data warehouses. The users of data warehouses are decision makers such as business planners, strategists and subject matter experts who issue queries such as illustrated in Table 8-1. As shown, the decision support queries require exploration and discovery of some sort.

As stated previously, there are fundamental differences between the users of data warehouse and operational applications (see Table 8-2). The most important difference is that the DSS users operate in a discovery mode - they look at one piece of information to learn what additional pieces of information will be needed. The operational users are typically involved in the day-to-day operational systems such as order processing, purchasing, shipping, and inventory control. Due to this reason they tend to update data frequently and require immediate response. In contrast, data warehouse users are usually managers and planners who browse through large quantities of data for ad hoc queries and forecasting applications that tolerate response delays.

Due to the differences in the users, the data characteristics of the decision support data are different from those of operational data (see Table 8-3). A data warehouse contains historical data that is summarized data, usually by subjects, for ease of queries. In many cases, the data warehouses are constructed from operational data by storing results of reports against operational data for later retrieval. In addition, data warehouses may contain additional information such as competitor data, and unstructured data such as proposals, meeting minutes, and forecasting newsletters. In contrast, operational databases contain detailed data that is accessed and updated frequently to support the day-to-day business operations. The operational data is accessed by hundreds, and in many cases thousands, of workers who need sub-second response as compared to the warehouse data that is usually queried by a few users who can tolerate response time in the range of 1 to 3 minutes.

Table 8-1: Examples of Operational Versus Decision Support Queries

Operational Queries	Decision Support Queries
• How much credit is available to Mr. Jones?	• Has Mr. Jones credit limit increased in the last three years?
• How many radios do we have on hand in the inventory?	• What was the best month for radio sales?
• How many claims did Dr. Dolittle file during the last month	• Which doctors filed the largest number of claims during the last year?
• What is the current salary of Sharon Walker?	• How many raises has Sharon Jones gotten since she joined this company?

Table 8-2: Operational Versus Decision Support Users

Operational Users	Decision Support (DW) Users
• Operate in predictable (non-discovery) mode	• Operate in discovery mode
• Line workers and clerical	• Decision makers and knowledge workers
• High concurrency requirements	• Low concurrency requirements
• Update data frequently	• Query data frequently
• Stringent response time requirements	• Relaxed response time requirements
• Access small amount of data	• Access large amounts of data

Table 8-3: Operational Versus Decision Support Data

Operational Data	Decision Support Data
• Atomic (detailed)	• Summarized
• Application-oriented	• Subject-oriented
• Current, accurate as of the moment of access	• Historical, represents values over time
• Dynamic (updated frequently)	• Static (updated infrequently)
• Transaction driven	• Analysis driven
• Data is updated in real-time (no periodic reloads)	• Data is reloaded or updated periodically

8.9.2 Basic Business Intelligence (BI) Tools

Business Intelligence (BI) concentrates on what really happened in the past and how this knowledge can be used to set future business directions. BI tools provide targeted information at the right place and time to improve the decision-making process. The objective is to deliver information that is actionable and enable companies to gain a competitive advantage in the highly competitive marketplace. Examples of the BI are the following:

- Predict changes in clothing industries based on changes in weather. For example, milder winters reduce the need for heavy jackets and overcoats.
- Use dashboards to understand which type of storms cause the most damage in what geographical areas
- Online food takeaway businesses using information about social events (e.g., school parties) to advertise and predict sales in targeted areas
- Use of dashboards to give a better and graphical view of sales opportunity pipeline and help managers by suggesting better allocations of sales resources.
- Use of dashboards to identify and focus on more profitable business investments
- Gambling businesses such as Harrah's Entertainment develops gambling profiles and identify most profitable customers

Principle tools used in BI include database query and reporting tools, executive information systems, online analytical processing (OLAP) and data mining. We will briefly discuss a few basic tools now and more later.

Data Query and Reporting Tools. Data query and reporting tools provide a wide range of capabilities for decision support analysts. Typical capabilities of these tools include:

- User screens that prompt users to type SQL statements and display results on the screen
- Point and click formulation of SQL statements.
- Query by Form (QBF) that displays a form that a user fills in. This form generates an SQL statement that is executed and displayed by the DBMS.
- Query by Example (QBE) that display tables that the user can fill in multiple rows with examples of the desired data.
- Reporting and display capabilities for hard copies as well as different type of display devices.

A large body of query tools, report writers, and 4GL tools at present provide these capabilities. A significant class of these tools allow users to build SQL queries by pointing and clicking on a list of tables and fields in the database. Most of these tools are targeted for "power" users who understand relational database concepts. Examples of tools in this category are the common RDBMS tools, Q+E from Q+E Software, and Data Prism from Brio.

Logical Viewing and Presentation Tools. These tools generate visualizations such as dashboards, bar charts and graphs for better understanding of the data. A simple example of a visualization tool is Excel that quickly generates many visualizations from spreadsheets. More sophisticated tools present a logical or business oriented view of data (e.g., customer object) and generate SQL queries based on operations on these objects. The results from these queries are also presented logically (sometimes through icons). In addition, these tools may hide the details of joins between tables from the end-users. Some tools in this category (e.g., Business Objects) present the views by creating end-user oriented aliases for tables and fields and then presenting these aliases through graphical objects. Other tools, such as MicroStrategy, may create meta data that identifies the contents and the location of the data in the warehouse.

Executive Information Systems (EIS) and Drill Down Tools. These tools provide analysis capabilities that are tailored for the executive decisions. A typical example of EIS is "drill-down" tools that allow executives to start at summary information and successively look at the details that make up the summary information. For example, an executive may first query a high level information item (e.g., the total travel expense for the entire organization in 2015). The executives then "drills down" to lower levels of details to understand what contributed to this expense. For example, if the travel expenses for 2015 were too high, then the executive would keep drilling down until the main department or individual who is travelling too much is found. In addition to drill down analysis, modern EIS tools offer analytical applications along functional lines such as financial analysis and sales, and allow ad hoc queries against multidimensional databases, and trend analysis and detection. Other examples of EIS are problem monitoring, and competitive analysis. Examples of EIS are Commissars Commander and Pilot Software's Lightship. The sidebar "Intelligent Executive Information Systems" shows the integration of artificial intelligence and Internet on EISs.

Intelligent Executive Information Systems. Developments in artificial intelligence and the Internet are being integrated into the executive information systems (EISs). The classic EISs have been based around drill down applications. However, EISs have been gradually integrating desktop tools, groupware, and data warehouses. Now the Internet and intelligent agents are finding their place in the EIS tools. The main trend is to replace a single drill down tool with a collection of Web-based tools that employ analytical as well as AI tools operating around data warehouses and the general Internet resources. For example, modern EISs can include intelligent, autonomous, software agents to search for changes in data of interest to executives and identify patterns. This data may be located in the data warehouse, operational databases, or the public Internet.

The emphasis of the intelligent agents for EISs should be more on organizational agents instead of personal agents, i.e., a collection of interacting agents that know about the organizational rules and policies. These agents should have the capabilities to filter through e-mail, organize the results of search on the Internet, monitor news findings, issue queries to the data warehouses, support dialogs with the executives, and exploit the capabilities of Internet to establish communications between groups of managers.

Case Study: Location Data Helps Shipping Company Map Out Business Route

Con-way Freight, a shipping and delivery company based in Ann Arbor, Michigan, uses geospatial data to make a significant impact on business processes. The company uses a location analytics initiative built around a combination of business intelligence (BI) and geographic information system (GIS) technology. Con-way did the following as part of their BI-GIS strategy:

- Created maps that include historical traffic patterns for a particular region and also added weather data to help develop more accurate delivery-time estimates. It also helped with planning for major weather events for different routes
- Mapped the locations of prospective customers to give sales managers a better idea of whether there is untapped business potential in a region
- Optimized delivery routes by making sure that drivers do not waste gas and time while making their deliveries.

- Overlayed customer data to GIS maps to discover patterns and trends

Con-way integrated GIS software from Esri with MicroStrategy-based BI reporting system. Although the integration was un-even, the company sees a great deal of business value in combining BI with GIS maps.

Source: Ed Burns, "Location analytics helps shipping company map out business route", http://searchbusinessanalytics.techtarget.com/feature/Location-analytics-helps-shipping-company-map-out-business-route, retrieved May 20, 2016

8.9.3 Business Analytics and Deep Learning

8.9.3.1 Business Analytics – Using Mathematical Techniques for Business Decisions

Data analytics is the use of *data, information technology, statistical analysis, quantitative methods, and mathematical models* to help managers gain improved insights about their business operations and make better decisions. There are many applications of business analytics (BA) in management of customer relationships, financial and marketing activities, supply chain management, human resource planning, and pricing decisions. BA has evolved from operations research and OLAP (Online Analytical Processing) into a profitable and revenue generation aspect of businesses. Business analytics consists of the following three basic techniques:

- *Descriptive Analytics*: uses data to understand and explore relationships between data items
- *Predictive Analytics*: analyzes past performance and predicts future results based on the past
- *Prescriptive Analytics*: uses optimization techniques to find the best (e.g., least expensive) solution

For example, BA is used in retail markdown decisions to clear seasonal inventory by reducing prices. The main question is: When to reduce the price and by how much? The following steps can be taken

- Descriptive analytics: examine historical data for similar products (prices, units sold, advertising, etc)
- Predictive analytics: predict sales based on price
- Prescriptive analytics: find the best sets of pricing and advertising to maximize sales revenue

One of the best known example of BA is Harrah's Entertainment that owns numerous hotels and casinos. Harrah's uses:

- Descriptive analytics to segment customers by gaming activities
- Predictive analytics to forecast demand for rooms
- Prescriptive models to set room rates, allocate rooms, and offer perks and rewards to customers

8.9.3.2 Deep Learning and Big Data

Data analytics of Big Data is a rapidly evolving area of work especially for business intelligence. For example, descriptive analytics uses Big Data to discover relationships between patient health and medication used, predictive analytics is used to predict the future based on past observations, and prescriptive analytics uses optimization techniques to find, say, the least expensive solutions to a problem. Many data analytics projects at present are based on Big Data repositories in health, education, public safety, public welfare and other vital sectors.

- Deep learning takes a model of human brain (called neural network) and populates it with Big Data on a subject matter (let us say skin cancer) to develop an expert system that is smarter than any skin cancer doctor. The basic premise of this work is that a human brain cannot store and retain extensive knowledge but a computer brain can. The idea is to develop smart lawyers and smart doctors as expert systems based on deep learning that are far better than human experts.
- Smart systems based on deep learning are arising raising several issues. However, the main advantage of smart systems based on deep learning is that these systems, once built, can: exchange information with all others. For example, if you are driving and learn about how to drive in heavy snow, then only you know it. But if an intelligent car learns the same lesson, then this knowledge can be replicated to all other cars quickly now and all other cars in the future.

8.9.3.3 Data Analytics Versus OLAP (Online Analytic Processing)

The term OLAP or Online Analytic Processing was coined by E.F. Codd to emphasize the analytical aspects of decision support tools [Codd 1993]. OLAP tools are optimized for dynamic, multidimensional analysis of consolidated enterprise data for analytical functions such as forecasting. OLAP became a very popular approach for business intelligence around 2000. Although OLAP is not very popular at present, it should be discussed briefly because it is closely related to business analytics. The main characteristics of OLAP tools are:

- Data drill down and data pivoting in multiple dimensions
- Complex analytical calculations that support descriptive, predictive and prescriptive techniques
- Immediate response through extensive use of online tools such as Web Browsers
- Multi-user, client/server environment

E.F. Codd has proposed 12 rules of OLAP (see the sidebar “The12 Rules of OLAP”). The most important aspects of these 12 rules is the emphasis on multidimensionality (rules 1, 6, 9, 12). To support the multidimensional analysis, multidimensional databases have been proposed. These databases extend the scope of relational databases (a two dimensional databases) to 3 dimensions and more. For example, the following three dimensions can be used for a sales database:

- Products
- Sales by month
- Sales by regions

A multi-dimensional database can efficiently store this information and allow three dimensional analysis of this information (e.g., show product sales by regions, show product sales by month, show product sales by month and by region, etc.). Although multi-dimensional databases are being used at present but they have been largely overtaken by Big Data technologies such as Hadoop.

The 12 Rules of OLAP (E.F. Codd)

1. Multidimensional conceptual view
2. Transparency
3. Accessibility
4. Consistent reporting performance
5. Client/server architecture
6. Generic dimensionality
7. Dynamic sparse matrix handling
8. Multiuser support
9. Unrestricted cross-dimensional calculations
10. Intuitive data manipulations
11. Flexible reporting
12. Unlimited dimensions/aggregation levels

8.9.4 Data Mining Overview

8.9.4.1 Data Mining – Discovering the Knowledge Hidden in Data

Data mining tools exploit a combination of AI and statistical analysis to discover information that is hidden or not apparent through typical query and analysis tools. Traditional database query tools are used to supply answers to simple questions such as "How many VCRs were sold in New York during December?". Analytics and OLAP (on-line analytical processing) tools go a step beyond the query tools and answer questions such as comparison of sales relative to targets by region for the last three years. Data mining steps go further by *finding patterns* and *inferring rules*. Specifically, the data mining tools use a variety of underlying technologies such as neural networks, decision trees, statistical analysis, and machine learning to detect:

- Associations (e.g., linking purchase of pizzas with beer),
- Sequences (e.g., tying events together such as marriage and purchase of furniture),
- Classifications (e.g., recognize patterns such as the attributes of customers who will discontinue doing business with you), and
- Forecasting (e.g., predicting future buying habits of customers based on past patterns).

Data mining has become a major growth area for marketing (especially for CRM) but has many other applications in healthcare, disaster recovery, transportation, agriculture, tourism and many other sectors. Although the basic mining algorithms have been around for several years, the availability of massive amount of corporate data and Big Data has provided a rich field for these tools to mine. Traditional data warehouses contain integrated, detailed, summarized, and historical data that can be mined to discover patterns of sales, customer buying habits, lifetime value (LTV) of customers, new customers likely to buy new products, demands on inventory, correlations between opening new stores and product sales, etc. However, Warehouses of Big Data go beyond the traditional corporate data warehouses to include large external resources such as social media, satellite feeds and micro data collected by sensors. The World Bank Open Data Initiative (http://datacatalog.worldbank.org/) is an excellent example of a Big Data Warehouse.

Although some capabilities provided by data mining can be provided by Analytics tools, the *distinguishing feature of data mining is knowledge discovery*, i.e., data mining arrives at insights automatically. For example, suppose that you have noticed a drop in the sales of your product and you want to know what is the cause of this. In most traditional analytical tools, you have to first form a hypothesis (e.g., product sales may be related to price increase), translate your hypothesis into queries, and interpret/understand the results of the queries. A data mining tool, on the other hand, will scan the data repositories and discover the correlations between product sale and other factors that you might have not thought about (sales might have dropped due to a competitor product, assuming this information is in the warehouse). The long range goal of data miners is to behave in a manner similar to "military intelligence" (i.e., gathering useful information from snapshots, notes, maps, observations).

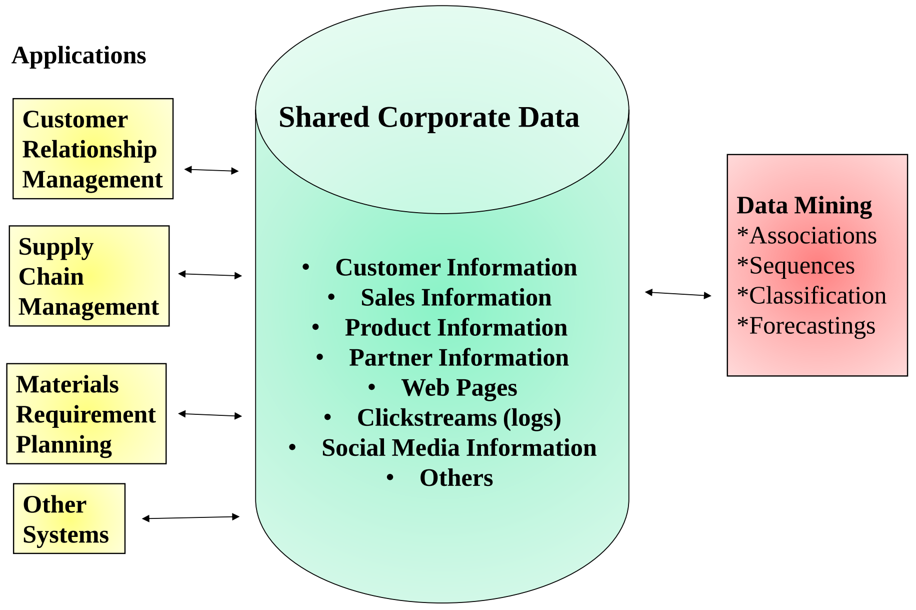

Figure 8-21: Data Mining in Enterprises

Key Characteristics of Data Mining:

- More discovery driven than Analytics
- Finds hidden patterns, relationships in large databases and infers rules to predict future behavior e.g., finding patterns in customer data for one-to-one marketing campaigns or to identify profitable customers.
- Types of information obtainable from data mining
    - Associations
    - Sequences

- Classification
- Clustering
- Forecasting

8.9.4.2 Data Mining Examples

Many examples of data mining are related to customer relationship management (CRM). For example, many data mining efforts have been devoted to understand the customer behavior to retain customers, acquire customers, upsell (i.e., sell more expensive products) and cross sell (sell related products). Why so many data mining tools are concerned with CRM? Let us consider the example of a wireless company that was adding 1.5 million customers / year but losing $900,000 customers/year. With average Revenue/Customer/Month=$55 , this amounted to losing $600,000,000 See the book "Building Data Mining Applications for CRM" by Berson et al, McGraw Hill, for a detailed discussion of this topic.

Data mining has also been used in direct mail (i.e., determine who is most likely to buy from direct mail), crime prevention (i.e., where crimes are most likely to happen), fraud detection (i.e., which insurance claims and credit card transactions are likely to be fraudulent), trend analysis (i.e., predict changes in customer behavior), and financial forecasts (i.e., models of stock markets).

Some examples of data mining queries are:

- What products the customer is likely to buy (based on current purchases)
- Which customers are likely to discontinue services
- What variables determine the customers who will go to a competitor
- Who is most likely to respond to mail
- How to detect fraudulent behavior in credit card users
- Which emarkets are likely to succeed

8.9.4.3 Issues with Data Mining

Data mining has several positive as well as negative aspects. Some of the plusses are:

- Tremendous value to organizations in supporting business strategies
- Data long forgotten is finding value ("hidden gold")
- Interesting applications of AI and statistics (many startups)

The possible minuses are:

- Privacy issues raised due to mining of credit card activities, medical history, past addresses and phone numbers, people you call, Web sites you visit, etc.
- Data ownership issues, i.e., who owns your data.
- A general feeling that there is increased reliance on past data instead of original and innovative thinking.

Due to the privacy issues, some attempts at regulations in US and Europe are underway. For example, companies cannot base models on gender, banks cannot give out credit history, etc.

8.10 Case Studies and Examples

8.10.1 Designing a Terrorist Screening Database

The FBI's Terrorist Screening Center, or TSC, was established after 9-11 to consolidate information about suspected terrorists from multiple government agencies into a single database -- The Terrorist Screening Database (TSDB). This database is the central terrorist watchlist that is used by multiple agencies to compile their specific watch-lists and for screening. In 2016, the list is estimated to contain over 2.5 million records. Approximately 5 percent of the people on the list are U.S. citizens or legal permanent residents.

The TSDB is included here because it is one of the most interesting, challenging and innovative example of designing a database, Big Data, and BI. According to the FBI site, the following statements illustrate how TSDB is supporting the fight against terrorism:

- Three of the 9-11 hijackers were stopped by state or local law enforcement for routine traffic violations in the days leading up to 9/11. At that time, however, no central system existed to identify them as having an association with terrorism.
- Today, state and local law enforcement officials, U.S. Customs and Border Patrol agents, Transportation Security Administration and State Department personnel, and other law enforcement officials have the TSC to help inform them if they are encountering a known or suspected terrorist. This information sharing has led to major improvements in counterterrorism efforts and public security, both at home and abroad. TSC makes terrorist identity information accessible through the National Crime Information Center system to more than 870,000 state and local officers nationwide

In essence, before the TSDB, multiple U.S. government agencies had been maintaining separate and inconsistent lists. Now, the consolidated database contains sensitive but unclassified information on terrorist identities, such as name and date of birth, that can be shared with other screening agencies. Classified information about the people in the watch list is maintained in other law enforcement and intelligence agency databases. Multiple agencies collect and maintain terrorist information and nominate individuals for inclusion in the TSDB consolidated watch list. They are required to follow strict procedures established by the head of the agency concerned and approved by the U.S. Attorney General. An individual will remain on the watch list until the respective department or agency that nominated that person to the list determines that the person should be removed from the list and deleted from the database

The TSDB is updated daily with new nominations, modifications to existing records, and deletions. The State Department system screens applicants for visas to enter the United States and U.S. residents applying for passports, while state and local law enforcement agencies use the FBI system to help with arrests, detentions, and other criminal justice activities. Each of these agencies receives the subset of data in the watch list that pertains to its specific mission. When an individual makes an airline reservation, arrives at a U.S. port of entry, applies for a U.S. visa, or is stopped by state or local police within the United States, the frontline screening agency or airline conducts a name-based search of the individual against the records from the terrorist watch list database.

When the computerized name-matching system generates a “hit” (a potential name match) against a watch list record, the airline or agency will review each potential match. Matches that are clearly positive or exact matches that are inconclusive (uncertain or difficult to verify) are referred to the applicable screening agency’s intelligence or operations center and to the TSC for closer examination. Despite extensive work, list’s usefulness needs to be optimized. Reports from both the Government Accountability Office and the Office of the Inspector General assert that the list contains inaccuracies and that government departmental policies for nomination and removal from the lists are not uniform.

While these selection criteria may be effective for tracking as many potential terrorists as possible, they also lead to many more erroneous entries on the list than if the process required more finely tuned information to add new entries. Notable examples of ‘false positives’ include Michael Hicks, an 8-year-old New Jersey Cub Scout who is continually stopped at the airport for additional screening and the late senator Ted Kennedy, who had been repeatedly delayed in the past because his name resembles an alias once used by a suspected terrorist. Like Kennedy, Hicks may have been added because his name is the same or similar to a different suspected terrorist. These incidents call attention to the quality and accuracy of the data in the TSC consolidated terrorist watch list.

Security officials say that mistakes such as the one that led to Anderson and Kennedy’s inclusion on no-fly and consolidated watch lists occur due to the matching of imperfect data in airline reservation systems with imperfect data on the watch lists. Many airlines don’t include gender, middle name, or date of birth in their reservations records, which increases the likelihood of false matches.

Many improvements have been made but this watch list and the others remain imperfect tools. There are numerous examples of people wrongly included in the list and known terrorists are not. The focus of TSDB has shifted more to mining Big Data especially the data generated by social media and cell phones. In addition, this database is used in collaboration with other government databases to infer and gain intelligence to combat terrorism.

Sources:

- FBI Site: **https://www.fbi.gov/about-us/nsb/tsc**
- FBI-Terrorist Screening Center-**https://www.fbi.gov/about-us/ten-years-after-the-fbi-since-9-11/just-the-facts-1/terrorist-screening-center**
- Laudon & Laudon, “Management Information Systems”, 12th Edition, Case Study (Chapter 6)

8.10.2 Short Case Studies

The following three case studies are extracted from the Information week website (www.informationweek.com/BigData).

8.10.2.1 How Patient Data Helps Hospitals

Washington State Health Care Authority has been able to reduce unnecessary use of the emergency room by adopting a system to exchange patient information electronically among emergency departments. The patient data that is gathered and distributed, helps hospitals identify frequent users and share information regarding their care.

Previously, the physician had no way of knowing which patients had visited multiple emergency rooms recently with the same complaint, or what diagnosis and treatment was given during prior visits. By using the data, emergency departments are now able to see all information about all emergency visits within the past twelve months and determine whether a patient is seeking narcotics or has a chronic condition, and then have the ability to respond accordingly.

Washington State Health Care has seen many benefits from this incorporation of big data (e.g., significant reduction of visits by frequent clients who visited five times or more annually). Other hospitals nationwide are looking for similar ways to incorporate big data and keep patients out of Emergency Departments, as well as better manage them when they do.

Source: http://www.hca.wa.gov/Documents/EmergencyDeptUtilization.pdf

8.10.2.2 How UPS Uses Big Data With Every Delivery

The different routes that can get the large number of UPS drivers from point A to point B are overwhelming. UPS engineers developed a plan for route-optimization that analyzed 200,000 possibilities for each route in real time. By using this big data, they were able to understand how vehicles performed during different routes and could easily see ways where deliveries could be improved.

One of the key business intelligence was the inefficiency of left turns -- UPS vehicles were wasting time and gas money from idling while waiting to make left turns. This led them to make one simple rule—to minimize, or if possible, eliminate left turns. Even though this meant drivers would often travel a greater distance, results showed that more packages could be delivered in less time with a reduced amount of emissions by driving in a series of right-hand loops.

As a result, between 2004 and 2012, UPS saved 10 million gallons of gas and carbon emissions were reduced by 100,000 metric tons (the equivalent of pulling 5,300 cars off the road annually). It also saved the company 98 million idle minutes or about $25 million worth of labor cost each year. In other words, this one simple change increased profits, met customer demands, improved safety and positively effected the environment.

Source: http://compass.ups.com/UPS-driver-avoid-left-turns/

8.10.2.3 Urban Waste Management Improves Its Age Old Service Using Big Data

The Greater Manchester Waste Disposal Authority (GMWDA), England's largest Waste Disposal Authority, has collaborated with the University of Manchester to enhance the area's environment by creating more sustainable solutions for the 1.1 million tons of waste produced there each year.

Like many other cities around the world, Manchester is looking to big data to help citizens better understand the importance of reducing waste. They are implementing government programs aimed to offer incentive to people who reduce waste or improve their waste management methods.

Following suit is Songdo, a futuristic "smart city" in South Korea, whose citizens use chip cards when disposing their garbage. The use of these cards enables Songdo's government to essentially measure how much waste is disposed of, as well as the time and location of the disposal. This provides data helpful for predicting the ideal moment for emptying containers as well as optimizing garbage collection routes.

Furthermore, researchers at the University of Stockholm are combining geographic and socioeconomic data to see where waste is spatially distributed and identify ways to better manage waste for the city. Using roughly half a million entries of waste fractions, locations and weights, these researchers were able to devise waste generation maps, which highlighted many areas for improvement.

Collecting trash may not seem like the most profound thing, but keeping it efficient is one of the most effective ways to improve our environment. Scientists expect that as the amount of data in this field gets bigger, the world's carbon footprint will get considerably smaller.

Source: https://datafloq.com/read/how-big-data-shapes-urban-waste-management-service/662

8.10.3 General Database and Data Warehouse Case Studies

Several short case studies and examples of databases, data warehouses, Big Data and BI appear regularly on the following sites:

- www.Informationweek.com
- www.eweek.com
- http://www.infoworld.com/
- http://www.cio.com/magazine
- http://www.cioinsight.com/
- Integrated Data Warehouse Case Studies, at Teradata website, http://www.teradata.com/Resources/Integrated-Data-Warehouse-Case-Studies/?LangType=1033&LangSelect=true
- Infosys Data Warehousing Case Studies, https://www.infosys.com/consulting/information-management/case-studies/Pages/data-warehousing-solutions.aspx

Time to Take a Break

Suggested Review Questions Before Proceeding

- What are data warehouses and how can they be used
- What is Hadoop and why is it needed
- What are the decision support and business intelligence (BI) tools and techniques
- What is data mining and how is it used to make management decisions
- What is the main idea of data analytics and deep learning

8.11 Chapter Summary

Data is a critical resource for enterprises because it is used to make business decisions and support almost all business applications. Different databases are used for different applications in modern enterprises. As stated previously, relational database technology is state of the market and state of the practice and SQL, the query language for relational DBMS has become a de-facto standard for enterprise-wide data access, even for non-relational data sources. However, relational DBMSs are not suitable for the Big Data applications that need to process massive amount of structured, semi-structured and unstructured data. Object-oriented DBMSs and NoSQL databases are being developed for Big Data at the time of this writing. In addition, we have briefly discussed data warehouses and the wide range of decision support tools to support BI, business analytics and data mining.

This chapter is intended to serve as a tutorial on how data is stored, organized and used for different applications in modern enterprises. Special emphasis has been is placed on the recent trends in Big Data and business intelligence. Specifically, we have attempted to answer the following questions:

- What are the basic data concepts
- What is enterprise (corporate) data and how it is evolving to Big Data
- What are Database Management System (DBMS)
- What are relational databases and what is SQL
- What are the basic principles of database design
- What are object-oriented and NoSQL databases
- What are data warehouses and how can they be used
- What is Hadoop and why is it needed
- What are the decision support and business intelligence (BI) tools and techniques
- What is data mining and how is it used to make management decisions
- What is the main idea of data analytics and deep learning

8.12 Major References

- Apache Software Foundation. (2013, 08 04). HDFS Architecture Guide. Retrieved from Hadoop: http://hadoop.apache.org/docs/stable1/hdfs_design.html
- Boyer, J. and Frank, B., "Business Intelligence Strategy: A Practical Guide for Achieving BI Excellence", Mc Press, 2010
- Date, C., "An Introduction to Database Systems", 8th edition, Addison Wesley, 2003
- Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. Retrieved from Google Research: http://static.googleusercontent.com/media/research.google.com/en/us/archive/mapreduce-osdi04.pdf
- Elmasri and Navathe, "Fundamentals of Database Systems", 7th Edition, Benjamin Cummings, 2015

- Foreman, J., "Data Smart: Using Data Science to Transform Information into Insight", Wiley, 2013
- Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google File System. Retrieved from Google Research: http://static.googleusercontent.com/media/research.google.com/en/us/archive/gfs-sosp2003.pdf
- Hernandez, M. "Database Design for Mere Mortals : A Hands-On Guide to Relational Databases" Addison-Wesley, 1997
- Kolb, J., "Business Intelligence in Plain Language: A Practical Guide to Data Mining and Business Analytics", CreateSpace Independent Publishing Platform, 2013
- Maheshwari, A., "Data Analytics Made Accessible", Amazon, 2014
- Martin, D., "Advanced Database Techniques", MIT Press
- Muller, J., "Database Design for Smarties: Using UML for Data Modeling", Morgan Kaufman, 1999
- Patil, DJ and Mason, H., "Data Driven", O'Reilly Media, 2015
- Provost, F and Fawcett, T., "Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking", O'Reilly Media, 2013
- Rajaraman, A., Leskovec, J., & Ullman, J. D., "Mining of Massive Datasets". Palo Alto, CA: Stanford University, 2013
- Sharda, R. and Delen, D., "Business Intelligence: A Managerial Perspective on Analytics", (3rd Edition), Pearson, 2013
- Sherman, R., "Business Intelligence Guidebook: From Data Integration to Analytics", Morgan Kaufmann, 2014
- Teorey, T.J., "Database Modeling and Design", Morgan Kaufman, 4th edition, 2005