# LO 1 - Describe the components and function of database management systems.

* 1. – Explain various terms related to database management.

Discussion: Why do we use database?

* To store information
  + User data
  + Product data for ecommerce sites
  + Business-related data.

### Terminology

**Database**

* Self-describing collection of integrated tables (Kroenke)
* A shared collection of logically related data and its descriptions, designed to meet the needs of an organization (Conally / Begg)
* An organized collection of data, today typically in digital form. The data are typically organized to model relevant aspects of reality, in a way that supports processes requiring this information (Wikipedia)

**Metadata**

* Data about data
* System catalog or data dictionary.
* Self-describing nature of a database provides program-data independence.

**DBMS – Database Management System**

* Examples of such systems include: Access, Oracle, SQLite, MySQL, SQL Server, DB2, PostgreSQL (postgres)
* A set of programs used to define, administer, and process the database and its applications.
* The software that manages and controls access to the database (Connolly/Begg)
* Translates the user’s data requests to the physical data storage.

**RDBMS -** Relational Database Management System

* DBMS that organizes data using relations (tables)
* Other DBMS techniques include (<http://en.wikipedia.org/wiki/Database> AND <http://en.wikipedia.org/wiki/Data_model>):
  + Flat Files
  + Hierarchical
  + Network databases
  + Object-relational

**Schema** – The entire structure of the database including all Tables, Attributes, Relationships, etc.

**Relation** – the definition of a table with columns (attributes) and rows (tuples) but not the data itself.

Example: Author Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **auID** | **auLName** | **auFName** | **Phone** | **Address** | **City** |
|  |  |  |  |  |  |

**Row, record, entry, tuple** – stores data about a specific item in your system

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **auID** | **auLName** | **auFName** | **Phone** | **Address** | **City** |
| 001 | Smith | John | 333-3333 | Box 123 | Saskatoon |
| 002 | Brown | Jane | 444-4444 | Box 321 | Saskatoon |

**Column, field, header, attribute** – describes the type of data that can be stored in that column for all the items in the table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **auID** | **auLName** | **auFName** | **Phone** | **Address** | **City** |

**Constraint** – restrictions on data in a table. Rules that keep data in a database as valid as possible. Example: the gender of a person can either be “M” or “F”. This is usually controlled by setting the domain for attributes in a table.

**Domain** – The set of allowable values for one or more attributes. A domain for an attribute can include criteria such as:

* the datatype (example: char, varchar, number, date),
* the size of the datatype (example: char(10)),
* whether the value is unique,
* whether the value exists in another table,
* whether the value has other constraints like business rules (**Check Constraint**). For example: An attribute can only be assigned a value between 0 and 9999.

**Superkey** – An attribute or a set of attributes that can uniquely id each row (tuple) in a table (relation). A superkey may contain additional attributes that are not necessary for uniquely identifying a row.

**Candidate Key** – Minimum attribute of a set of attributes. This means that there is no subset (smaller set) of the candidate key that uniquely identify each row in a table. A table may have several candidate keys.

**Primary Key** – one of the candidate keys is selected to be the primary key. Does not allow nulls.

**Foreign Key** – a primary key or unique key from a parent table that exists in a child table for the purposes of linking the tables together.

**Composite Key** – a key made up of more than one attribute

**Unique Key** – An attribute or set of attribute that is unique in a table (no two records can have the same value). A unique key allows nulls.

**Surrogate Key** – A value that is generated to uniquely identify a row. It is not derived from any application data. Usually created / used to act as a primary key and is usually a sequential number.

Question: What’s the difference between a Candidate Key and a Composite Key? A candidate key is any possible option that could be used as a primary key but does not have to be the primary key. A composite key is the actual primary key if the primary key is made up of more than 1 attribute.

**SQL** – Structured Query Language. Most commonly used language for interacting with a relational database.

**Components of SQL**

* **DDL** – Data Definition Language
  + Commands used to specify the database schema (ex: CREATE)
* **DML** – Data Manipulation Language
  + Read and update the database (SELECT, INSERT, UPDATE, DELETE)
* **DCL** – Data Control Language
  + Control permissions in the database (GRANT, REVOKE)

**4GL** – Fourth Generation Programming Language (basically shorthand programming that requires much fewer lines of code than a 3GL)

* Non-procedural – concerned with “what” to do and **NOT** “how” to do it. Instead of defining steps for performing a task, the programmer will define the parameters for the tool.

**Transaction –** A unit of work comprised of a number of different steps. Data will be unreliable if any of the steps in the transaction are missed. Example: withdrawing money from a bank machine involves: 1. Looking to ensure there is enough cash in the account. 2. Debiting the account 3. Providing actual cash to the customer. If any of these steps fail, the “transaction” is incomplete and the data is incorrect.

**Data mining** – analyzing collected data (often stored in a data warehouse) for patterns.

**Big data** – Collection of large and complex data sets to which data mining is usually applied.

## 1.2 Discuss the historical development of DBMSs and the relational model and other database models

<http://www.computerhistory.org/revolution/memory-storage/8/265/2207>

The first crack at database management was the File-based systems which emulated manual filing systems.

**File-based systems**

* First type of digital database system.
* Emulated manual filing systems.

So by a manual file system we’re referring to how we would take a physical file folder and set up these files with all the external and internal correspondence relating to an item (employee, client, project, product, etc). Backup would be having multiple copies of the file in different physical locations and security was locking doors and/or filing cabinets. While this system works well when we only have to store and retrieve information, it doesn’t work well when we need to cross-reference or process information.

Let’s take a look at an example. A real estate office could have a separate file for each property for sale or rent, each seller, each potential buyer, each renter, and each employee. Let’s assume we are using the file-based approach:

What information/data would be easy to find (example)?

* Create a file for a new employee (Karen).
* Create a file for a new property available for rent.
* Find the salary information for **Bob**
* Find what year **Jane** bought a house.
* Find out how many properties **Jim** has rented.

What information / data would be hard to find (examples)?

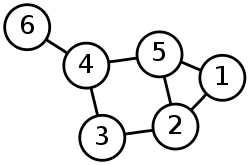
* Find all the properties with 3 bedrooms that is for sale.
* Find all properties that are within 3 miles of the city centre
* Calculate the average rent for a 2-bedroom Condo
* Calculate the total annual expenditure on employee salaries.

**Limitations of the File-Based Approach**

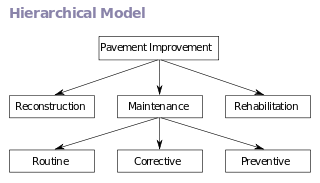
* Separation and isolation of data
  + Since data was kept in separate files, it was difficult to get information out because significant processing was needed to derive the information
* Duplication of data
  + Often file-based systems are built on a department basis, so data is duplicated between departments.
  + Duplication often results in wasted time, space, loss of data integrity.
* Data dependence
  + Physical structure and storage of the data files is defined by the application. Changing the structure requires a lot of work.
* Incompatible file formats
  + Structure of the data is dependent on the program and programming language.
* Fixed queries
  + The information that can be derived is dependent on the application developers writing code to develop the reports. There is no facility to do ad-hoc reporting.

To solve the limitations of filebased systems the Database concept started in the 1960s.

The first generation of database systems were navigational , where applications accessed data by following pointers from one record to another. Ie. Node6.Node4.Node5.Node1

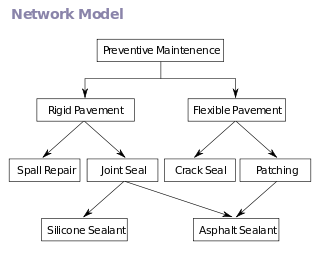


There were two main types of navigational databases, the network model (Codasyl a consortium model first proposed by Charles Bachman) and the hierarchical (IBM IMS system) model.

**The hierarchical model**

* **The hierarchical model is tree-based.**
* Each record has one parent and potentially many children

**The network model**



* Is graph based.
* Allows for each record to have multiple parents and child records.

These systems were used until the 1980s where they were eventually replaced with relational databases. There was a resurgence of the hierarchical data systems in the late 1990s with the introduction of XML based systems.

**Relational Database Model**

The relational database model was proposed in 1970 by Edgar Codd. This model departed from the navigational model.

Relational database Models:

* Specified that applications search for data by the content of the data (not by following links).
* This allows the content of the database to evolve without rewriting applications.
* Primary and foreign keys are used instead of pointers to define relationships between entities.

The use of primary and foreign keys made for a new paradigm in which searches could be conducted leading to a new language for searching relational databased, called Structured Query Language (SQL).

While the relational database is a good model for searching for content, it is less well suited to modeling the data (giving the person a visual of what the data looks like). For this purpose, a different model was proposed in 1976 called the entity-relationship model (a type of diagramming). This model is now the primary form for modeling databases today.

*Advantages of DBMS*

* Controls data redundancy (valid normalization)
* Improves data integrity (validity) / consistency
* Increase productivity the dbms provides all the low-level file-handling routines.
* Improved ability to share data
* Improved security

*Disadvantages*

* Complexity
* Cost of software
* Higher impact of a failure.
* Scalability (over multiple servers) \*though MySQL has just comes out with it’s “Fibre” product which is supposed to allow scaling tables over multiple servers.

The relational model is rigid. In the relational model, all data is held in tables with a fixed structure of rows and columns. This is becoming restrictive so new types of data are moving into mainstream use. For example, databases that are used to store documents, engineering information (CAD models) or molecular science (3d molecule models and meta information) do not easilty fit into the relational model. To address this problem, three new models are worth a note, the object database, the XML database, and NoSQL databases.

**Object Oriented Database**

Object oriented Databases were first define in 1985 and allows for information to be represented in the form of objects as used in object oriented programming. Most object databases offer some kind of query language that generally makes use of OQL (Object Query Language). Since object database store both the complex data and the relationship between data directly, without have to map to row or columns, this allows them to work well with very complex data.

Potential advantages

* Objects don’t require assembly and disassembly saving coding time and execution time.
* Reduces paging.
* Easier navigation
* Better concurrency control – a hierarchy of objects may be locked.
* Data model is based upon the real world.
* Works well for distributed architectures.
* Less code required when applications are object oriented.

Disadvantages

* Lower efficiency when data is simple and relationships are simple
* Relational tables are simpler
* Access speeds may be slower
* More user tools for RDBMS
* Standards for RDBMS are more stable
* Support for RDMS is more certain and change is less likely to be required.

**XML Database**

XML database is a data base that allows for data to be stored in XML format. The data can then be queried, exported, and serialized into the desired format. These databases are usually associated with document-oriented databases. The major reason for this type of database is that is becoming increasingly popular for data transport to make use of XML. This means that data is extracted from databases and put into XML documents and vice-versa.

**NoSQL database**

NoSQL (Not Only SQL) is the current trend mostly being utilized by the web development world. NoSQL is used to describe the new datastores that are non-relational. This should NOT be confused with No SQL as some NoSQL platforms use a type of SQL. This trend started because up until recently RDBMS could not scale out (be distributed across multiple servers) so the only way to have a bigger database was to scale up (add more hardware power to your single database server). Now, it is usually more cost effective to scale out rather than scale up so they came with NoSQL which could scale out more easily. There are many different ways of classifying NoSQL databases such as by data model or feature., and right now there is no agreement in the community on what the classification should be) but it seems that there are 4 categories. The 4 basic categories are:

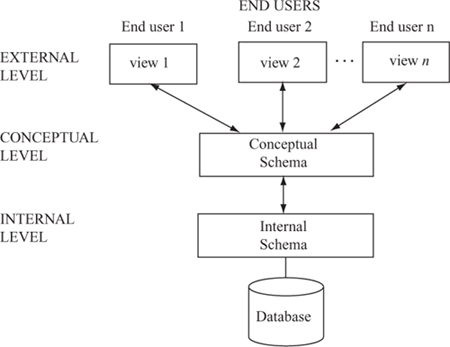
* Column – basically an object that consists of 3 elements (unique name, value, timestamp) (Google’s Bigtable) (storing/processing in a column instead of a row)
* Document – documents encapsulate and encode our data. Example: XML, Apache CouchDB
* Key-value – basically a map and a dictionary that allows clients to put and request values per key. (Amazon’s Dynamo)
* Graph – A type of navigational database that employs nodes, properties, and edges where the node is your “entity” (Oracle NoSQL database)

## Diagram the architecture of a DBMS

When talking about DBMS architecture, we are referring to how the database is viewed by users. Abstract views of the data are provided to the users. These views hide how the data is stored so that the user does not need to worry about where data is located or how it’s stored.

The architecture is comprised of 3 different levels: external, conceptual and internal.

* External: The highest level of abstraction, often referred to as the view level. Defines how each end-user understands the organization of the relevant data within the database. A single user can have a number of different views at the external level. A security feature of the database that is hidden from users.
* Conceptual: This level of abstraction deals with the logical structure of the entire database and thus, is also known as the **logical level**. It describes what data is stored in the database, the relationships among the data and complete view of the user’s requirements without any concern for the physical implementation. It hides complexity of physical storage structures. It unifies the various external views into a coherent global view. It comprises all the end-user needed generic data. IE: All the data from which any view may be derived or computed.
* Internal (Physical Level): It’s the lowest level of abstraction that deals with the physical representation of the database on the computer and thus, is also known as the **physical level**. It describes *how* the data is store physically and organized on the storage medium. Resides inside the DBMS. It is concerned with the operational matters of the database. It deals with the storage layout of the conceptual level, provides support storage structures like indexes in order to enhance performance, and occasionally stores data of individual **views**.



DBAs may have certain details

Developers/DBAs

End Users

Major components of DBMS architecture:

* **DBMS External Interfaces** – Means of communicating with the database to perform operations on a database and manage the DBMS.
  + Direct database operations: defining data types, assigning security levels, etc.
* **Database Language Engines (or processors)** – Most operations upon database are performed through expression in Database languages. Languages exist for data definition, data manipulation and queries (SQL), as well as for specifying various aspect of security. Language expressions are fed into DBMS through proper interfaces. A language engine processes the language expressions (via a compiler or interpreter) to extract the intended database operations in a way that they can be executed on the DBMS.
* **Query Optimizer** – Performs query optimization. Attempts to determine the most efficient way to execute a given query by using the most efficient query plan.
* **Database Engine** – core service for storing, processing, and securing data.
* **Storage Engine** – handles the SQL operations for different table types.
* **Transaction Engine** – For correctness and reliability purposes most DBMS internal operations are performed encapsulated in transactions. Transactions can also be specified externally to the DBMS to encapsulate a group of operations.

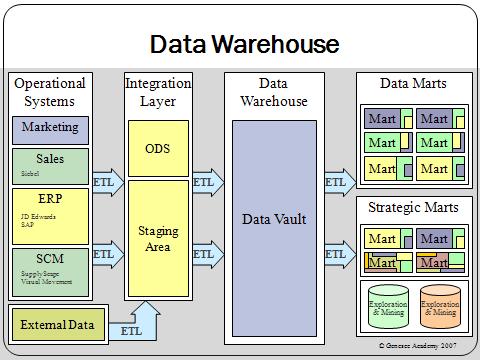
## Discuss data warehouses and multidimensional data structures

### <https://www.youtube.com/watch?v=zTs5zjSXnvs>

### Data Warehouse

Data warehouse (DW) – is a database used for reporting and data analysis to assist management in decision making. It is usually data archives that are taken from different enterprise application systems.

The data warehouse database tends to be:

1. Subject-oriented - organized around subjects (customers, products, sales) instead of major functional areas (invoicing, stock control, product sales)
2. Integrated – integrating multiple data sources to be consistent.
3. Time variant – data represents a series of snapshots and is only valid at some point in time or some interval. (example: the change in price of a product over time)
4. Non-volatile – data is not added in real time but instead refreshed on a regular basis. Data is added as a supplement and does NOT replace old data. Batch processes. For [](http://upload.wikimedia.org/wikipedia/commons/4/46/Data_warehouse_overview.JPG)

For example, every Friday night at 11:30 PM a batch process runs to export data from the Marketing department (their own databases in 3NF), performs some ETL on that data and inserts into the Data Vault.

The **Extract, Transform and Load (ETL)** process:

* Extracts data from various operational systems
* Transforms the data to fit operational needs within the integration layer.
* Loads the data into the data warehouse.

These pieces of information are then stored in the Marts (Data and Strategic). Marts are used to improve performance and ease of use within that specific area. They are the access layer for providing data to the users.

Data mart – a subset of the DW – it’s own small data warehouse focused on a specific area of interest.

Strategic mart – data warehouse focused on data that will be used for determining business direction and setting strategic goals. (periodic trend analysis, data used for risk analysis)

The data is often grouped into hierarchical groups (trees) called *dimensions* and also into facts and aggregate facts. The combination of facts and dimensions is sometimes called a star schema.

There are two main approaches to storing data in a data warehouse, the dimensional approach (Star Schema) and the normalized approach. Do you remember normalization? Not particularly, it was covered a little in September last year. Normalization is essentially rules for preventing update anomalies and data inconsistencies.

In a dimensional approach, transaction data are partitioned into "facts", which are generally numeric transaction data, and "dimensions", which are the reference information that gives context to the facts.

Example: A sales transaction can be broken up into

1. facts:
   1. the number of products ordered
   2. the price paid for the products
2. and dimensions:
   1. order date
   2. customer name
   3. product number
   4. order ship-to and bill-to locations
   5. salesperson responsible for receiving the order.

The main disadvantages of the dimensional approach are:

1. In order to maintain data integrity of facts and dimensions, loading the data warehouse with data from different operational systems is complicated

2. It is difficult to modify the data warehouse structure if the organization changes the way in which it does business.

It should be noted that both normalized – and dimensional models can be represented in entity-relationship diagrams as both contain joined relational tables. The difference between the two models is the degree of normalization.

### Multidimensional Data Structures

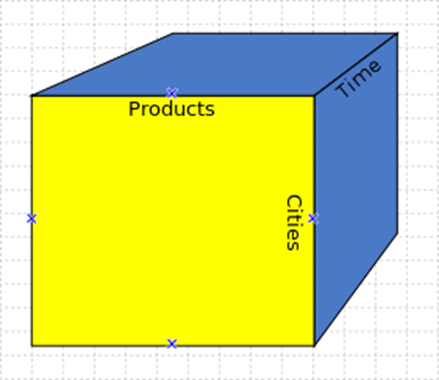
<https://www.youtube.com/watch?v=2ryG3Jy6eIY>

In Relational databases, tables are only able to provide a 2 dimensional view of data. Sometimes it’s necessary to compare more than 2 dimensions. The multidimensional structure is similar to the relational model. The dimensions of the cube-like model have data relating to elements in each cell. This structure gives a spreadsheet-like view of data. This structure is easy to maintain because records are stored as fundamental attributes—in the same way they are viewed—and the structure is easy to understand. Its high performance has made it the most popular database structure when it comes to enabling online analytical processing (OLAP).

A cube can be thought of as a generalization of a two-dimensional spreadsheet. For example, a company might wish to summarize financial data by product, by time-period, by city to compare actual and budget expenses. Product, time, city and scenario (actual and budget) are the data’s dimensions.

The cube is a shortcut for multidimensional datasets, given that data can have a arbitrary number of dimensions. The term “hypercube” is sometimes used, especially for data with more than three dimensions.

Each cell of the cube holds a number that represents some measure of the business, such as sales, profits, expenses, budget, and forecast.



#### OnLine Analytical Processing

A cube can be thought of as a generalization of a two-dimensional spreadsheet. For example a company might wish to summarize financial data by product, by time-period, by city to compare actual and budget expenses. Product, time, city and scenario (actual and budget) are the data's dimensions.

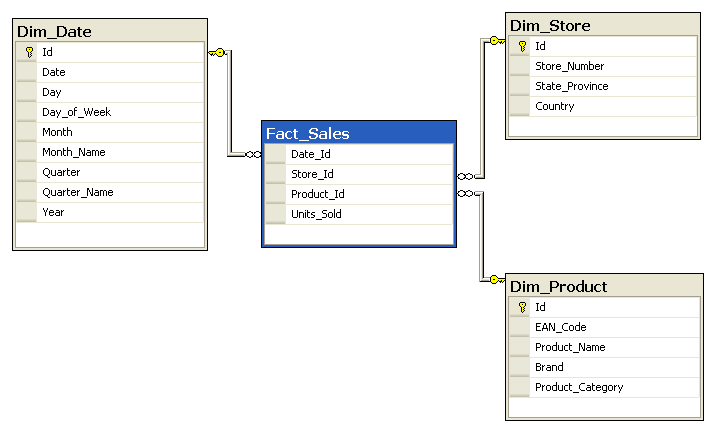
Cube is a shortcut for multidimensional dataset, given that data can have an arbitrary number of dimensions. The term hypercube is sometimes used, especially for data with more than three dimensions.

Each cell of the cube holds a number that represents some measure of the business, such as sales, profits, expenses, budget and forecast.

OLAP data is typically stored in a star schema or snowflake schema in a relational data warehouse or in a special-purpose data management system. Measures are derived from the records in the fact table and dimensions are derived from the dimension tables.These types of schemas are used for data warehouses and data marts where the speed of data retrieval is more important than efficiently manipulating the data. Often they are designed just short of third normal form.

**Star Schema**

So let’s think about this – the measure is our sales ie) on date id 5, store 5 sold 10 of product 8. (We can get this from our Fact sales table). Then, if we require any other dimensions, we can look at the dimension tables, ie: what product? Look at the product name and we can find out we sold 10 widgets. We’d likely want to look at the bigger picture information so that we could tell things like increases in sales of a product, if there are particular events that cause an increase/decrease in items etc.

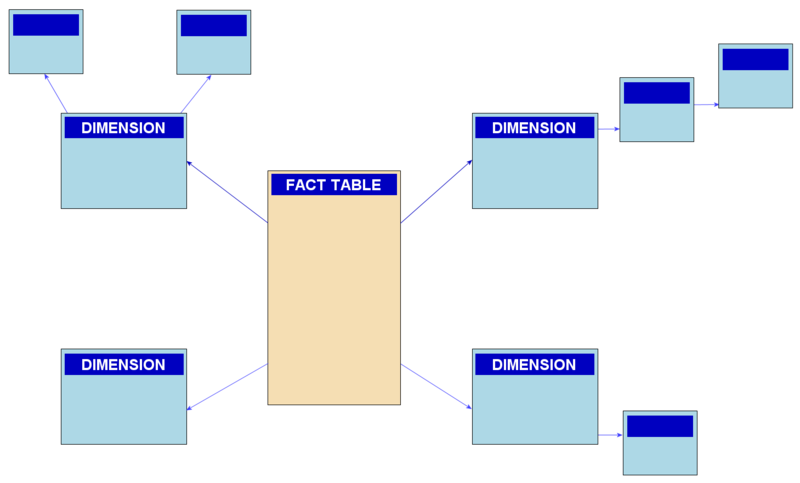


De-normalized data is used. In the star model, dimensions directly refer to fact table and business hierarchy. It is not implemented via referential integrity between dimensions. Now, remember from the DW video, it talked about a term called “Constellation”, in that example you could see that you would have multiple dimension tables that may hold similar data.

**Graphical user interface, text, application

Description automatically generated**

**Snowflake Schema**

[](http://upload.wikimedia.org/wikipedia/commons/b/b2/Snowflake-schema.png)

Data is organized for eliminating redundancy which reduces the amount of data we have. The hierarchy of the business and its dimensions are preserved in the data model through referential integrity.

**Review for LO1:**

Work on review questions document.

Optional Resources and Extra Information:

Data mining – analyzing collected data (often stored in a data warehouse) for patterns.

Data mining video (Viewer discretion advised - this video uses language that some may find offensive. It also shows a dramatization of a pregnant woman’s water breaking). <https://www.youtube.com/watch?v=f2Kji24833Y>

Big data – collection of large and complex data sets to which data mining is usually applied.

Ted talk 22 minutes on big data: <https://www.youtube.com/watch?v=Zr02fMBfuRA>