



# coding-bootcamps.com

Introduction to Database Design

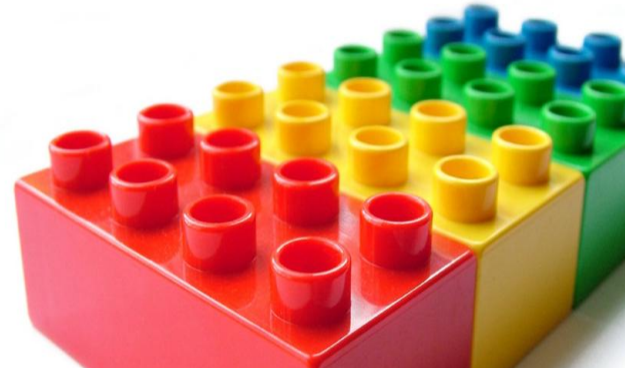


# Coding Bootcamps

By Jace Fugate from [Coding Bootcamps](#)

# Prerequisite

None



# Before the Advent of Database Systems

Session 1

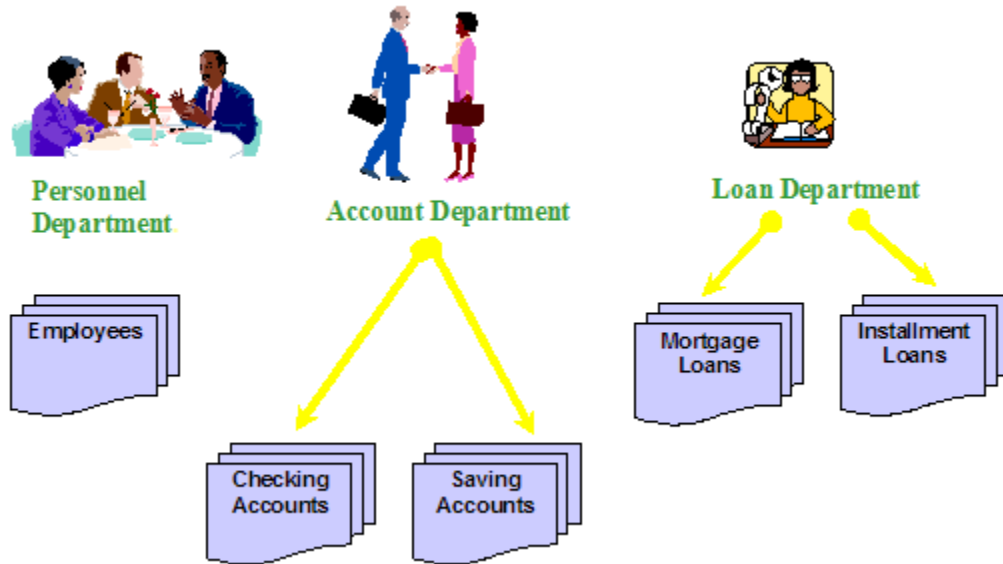
# Session 1

- File-based Approach
- Database Approach



# File-based System

- One way to keep information on a computer is to store it in permanent files.



## Disadvantages of the file-based approach

**Data Redundancy:** Since files and applications are created by different programmer of various departments over long period of time, it might lead to several problems:

- Inconsistency in data format
- The same information may be kept in several different place (files).
- Data inconsistency which means various copies of the same data are conflicting ; waste storage space and duplication of effort

# Disadvantages of the file-based approach

## **Data Isolation**

It is difficult for new application to retrieve the appropriate data which might be stored in various files.

## **Integrity problems**

- Data values must satisfy certain consistency constraints which are specified in the application programs.
- It is difficult to add change the programs to enforce new constraint

## Disadvantages of the file-based approach

### **Security problems**

- There are constraint regarding accessing privileges
- Application is added to the system in the ad-hoc manner so it is difficult to enforce those constraints

### **Concurrent** – access anomalies

Data may be accessed by many applications that have not been coordinated previously so it is not easy to provide a strategy to support multiple users to update data simultaneously



## Database approach

Database and database technology play an important role in most of social areas where computer are used, including business, education, medicine etc. To understand the fundamental of database system, we start from introducing the basic concepts in this area.

Database is a shared collection of related data which will be used to support the activities of particular organization. Database can be viewed as a repository of data that is defined once and then is accessed by various users.

## The meaning of data

Data are factual information such as measurements or statistics about objects and concepts. We use data for discussions or as part of a calculation. Data can be a person, a place, an event, an action or any one of a number of things. A single fact is an element of data, or a *data element*.

# Fundamental Concepts

Session 2

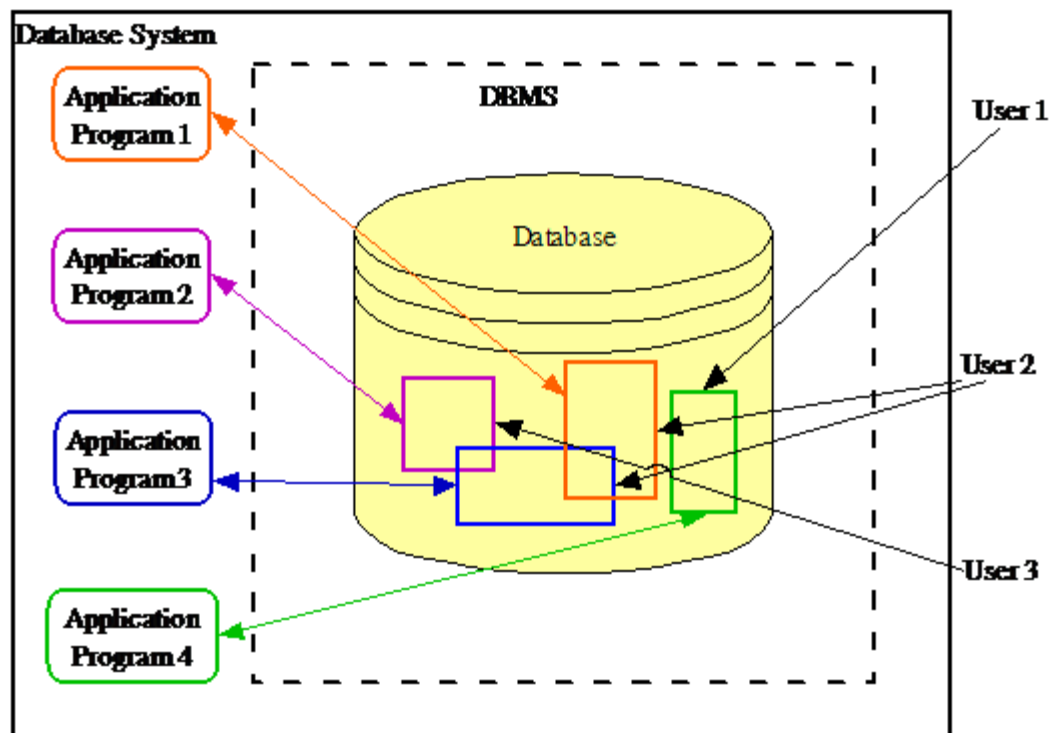
# Session 2

- What Is a Database?
- Database Management System

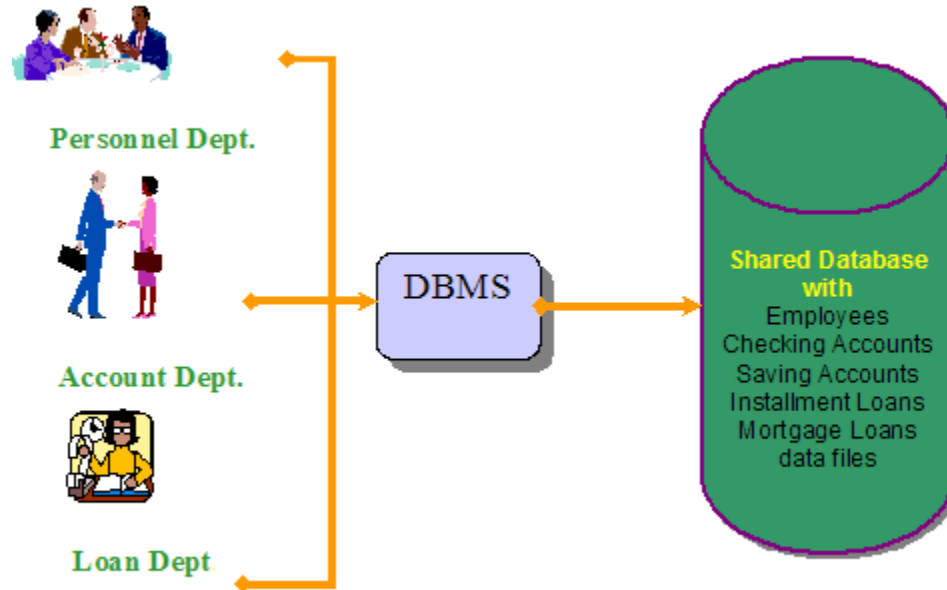


## A database has the following properties:

- It is a representation of some aspect of the real world; or perhaps, a collection of data elements (facts) representing real world information.
- Database is logical coherent and internally consistent.
- Database is designed, built, and populated with data for a specific purpose.



# Database management system



# Characteristics and Benefits of a Database

Session 3



# Session 3

- Characteristics and Benefits of a Database



## Why database?

We often need to access and re-sort data for various uses. These may include:

- Creating mailing lists
- Writing management reports
- Generating lists of selected news stories
- Identifying various client needs

## Why database?

The processing power of a database allows it to manipulate the data it houses, so it can:

- Sort
- Match
- Link
- Aggregate
- Skip fields
- Calculate
- Arrange

## Why database?

Because of the versatility of databases, we find them powering all sorts of projects. A database can be linked to:

- A website that is capturing registered users
- A client-tracking application for social service organizations
- A medical record system for a health care facility
- Your personal address course in your email client
- A collection of word-processed documents
- A system that issues airline reservations

# Characteristics of database approach

## Self-Describing Nature of a Database System:

Database System contains not only the database itself but also the descriptions of data structure and constraints (meta-data). These information is used by the DBMS software or database users if needed.

This separation makes database system totally different from traditional file-based system in which data definition is a part of application programs

## Characteristics of database approach

Insulation between Program and Data:

In the filed base system, the structure of the data files is defined in the application programs so if user want to change the structure of a file, all the programs access to that files might need to be changed. On the other hand, in database approach, data structure is stored in the system catalog not in the programs so such changes might not occurs.

## Characteristics of database approach

**Support multiple views of data:** A view is a subset of the database which is defined and dedicated for particular users of the system. Multiple users in the system might have different views of the system. Each view might contains only the interested data of an user or a group of user.

**Sharing of data and Multiuser system:** A multiuser database system must allow multiple users access the database at the same time. As the result, the multiuser DBMS must have concurrency control strategies to ensure that several user try to access the same data item at a time do so in the manner so that the data always be correct.

# Benefits of Database Approach

## **To control Data Redundancy**

- In the Database approach, ideally each data item is stored in only one place in the database
- However, in some case redundancy is still exists to improving system performance, but such redundancy is controlled and kept to minimum

## **Data Sharing**

The integration of the whole data in an organization leads to the ability to produce more information from a given amount of data



# Benefits of Database Approach

## **Enforcing Integrity Constraints**

DBMSs should provide capabilities to define and enforce certain constraints such as data type, data uniqueness.

## **Restricting Unauthorized Access**

- Not all users of the system have the same accessing privileges.
- DBMSs should provide a security subsystem to create and control the user accounts.

# Benefits of Database Approach

## Data Independence

- The system data descriptions are separated from the application programs.
- Changes to the data structure is handled by the DBMS and not embedded in the program.

## Transaction Processing

The DBMS must include concurrency control subsystem to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.

# Benefits of Database Approach

## **Providing multiple views of data**

- A view may be a subset of the database. Various users may have different views of the database itself.
- Users may not need to be aware of how and where the data they refer to is stored

## **Providing backup and recovery facilities**

If the computer system fails in the middle of a complex update program, the recovery subsystem is responsible for making sure that the database is restored to the stage it was in before the program started executing.

# Types of Data Models

Session 4

# Session 4

- High-level Conceptual Data Models
- Record-based Logical Data Models



## Data models

Data Model is a collection of concepts that can be used to describe the structure of database. Structure of database means data types, relationships and constraints.

In addition, most data model include a set of basic operations for specifying retrievals and modifications on the database.

## Data models

Data Model provides a means to achieve **Data Abstraction**.

Data Abstraction is refers to the hiding of certain details of how the data are stored and maintained. With several levels of abstraction, the user's view of the database is simplified and this leads to the improved understanding of data.

We discuss Data Abstraction in depth in the next session.

# Types of Data models

**High-level Conceptual Data Models**

**Record-based Logical Data Models**



# Types of Data models

## High-level Conceptual Data Models

Provide concepts that are close to the way people perceive data to present the data. Typical example of this type is entity – relationship model which use main concepts like entities, attributes, relationships. An entity represents real-world object such as an employee, a project. An entity has some attributes which represents properties of entity such as employee's name, address, birthdate. A relationship represents association among entities for example a works on relationships between employee and project.

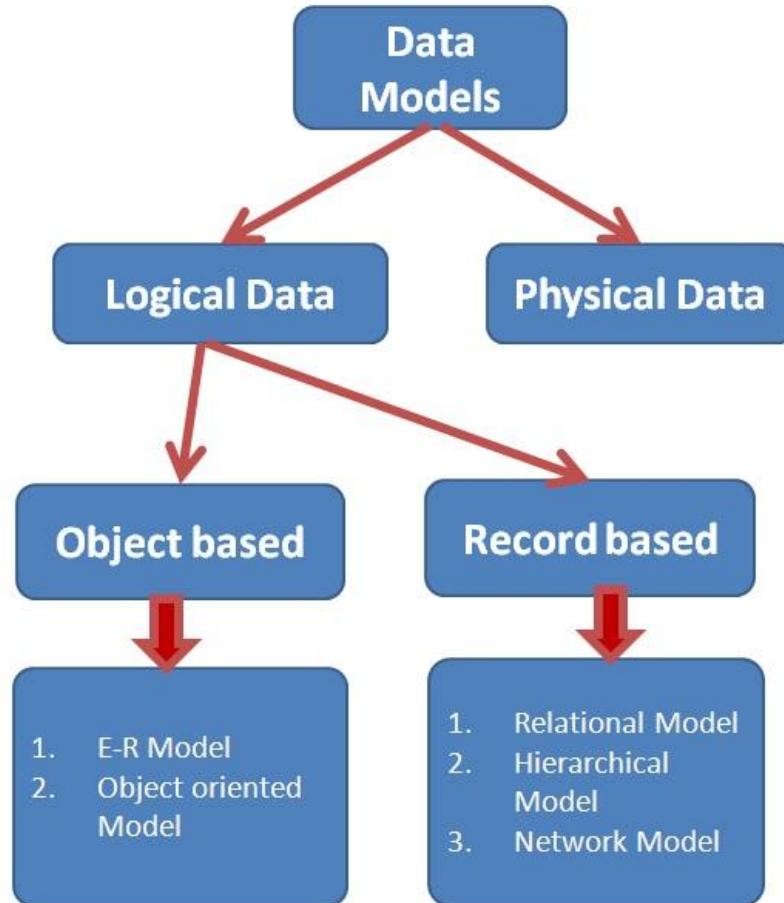
# Types of Data models

## **Record-based Logical Data Models**

Provide concepts that can be understood by the user but not too far from the way data is stored in the computer. Three well-known data models of this type are relational data model, network data model and hierarchical data model.

## Types of Data models

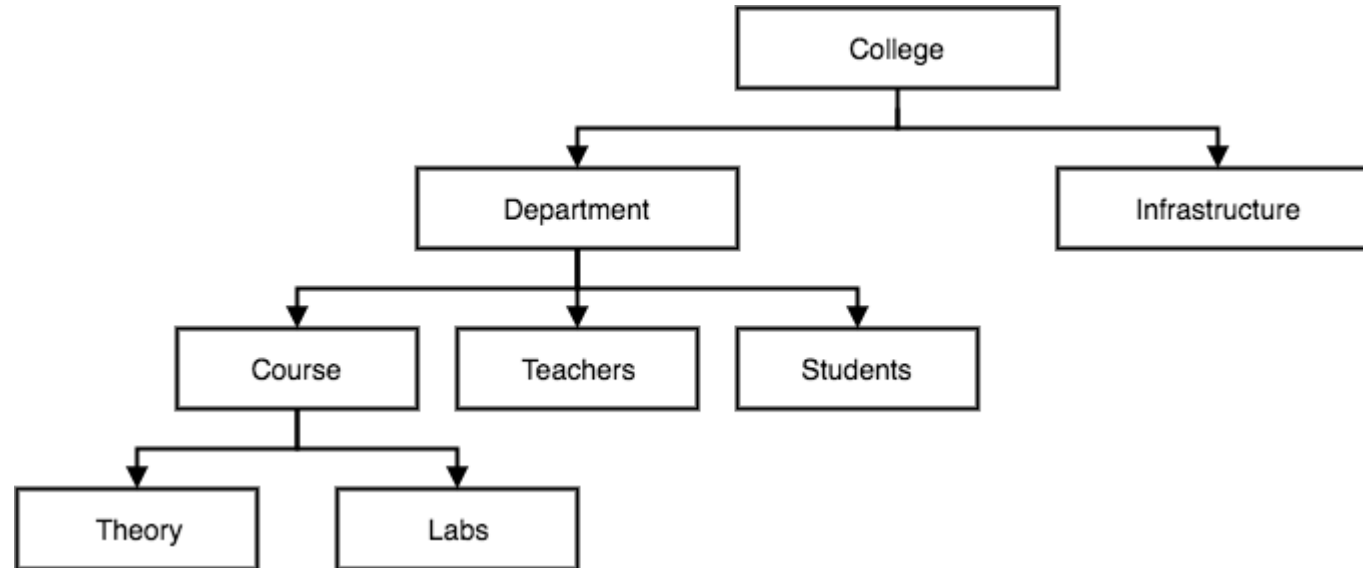
We will discuss each type in depth in next sessions



# Types of Data models

For more details, read your course materials

## Hierarchical data tree structure



# Data Modeling

## Session 5

# Session 5

- Degrees of Data Abstraction
- Data Abstraction Layer
- Schemas
- Logical and Physical Data Independence



There are three levels of abstractions:

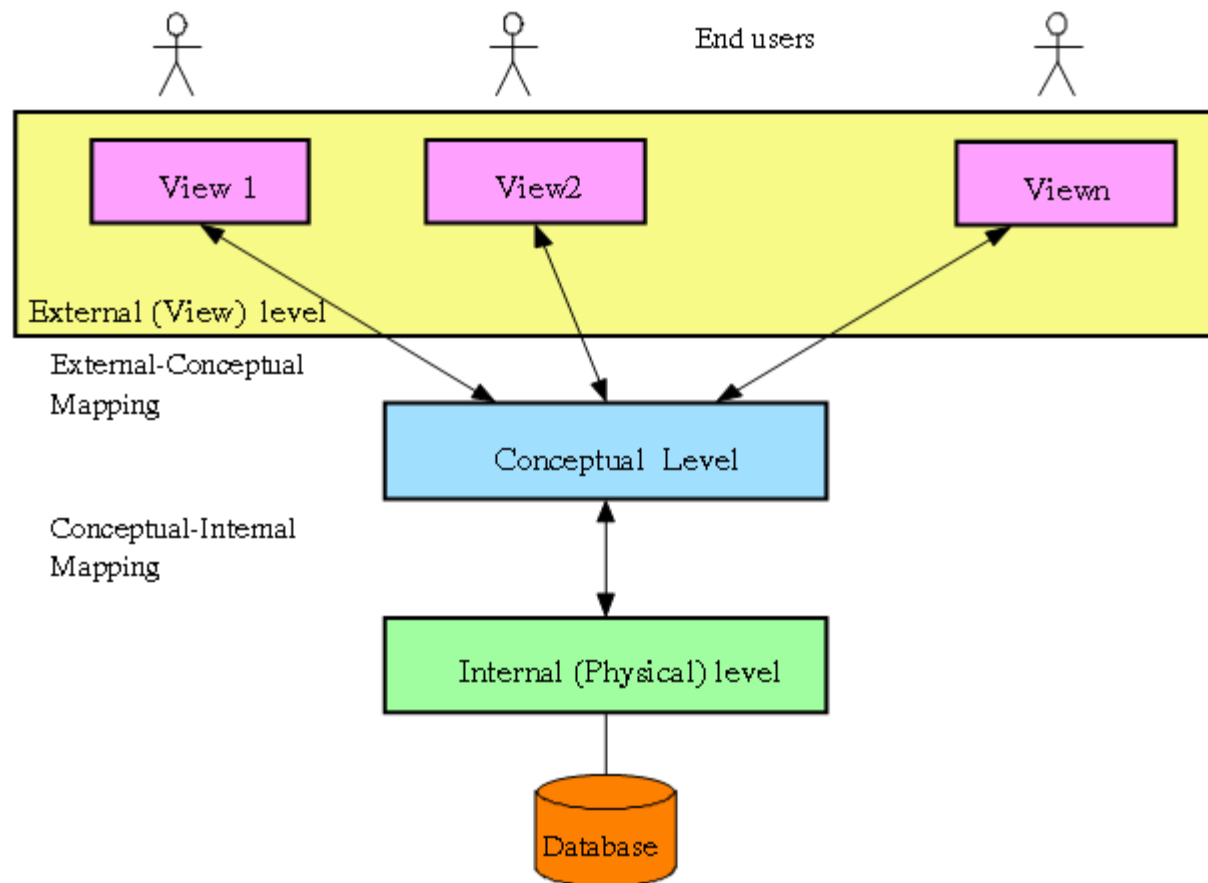
**1- View level:** The highest level of abstraction describes only part of the entire database. Many users will not be concerned with the large database. Instead, they need to access only a part of it so that view level abstraction is defined. There are many views for the same database.

**2- Logical level:** This level describes what data are stored in the whole database.

**3- Physical level:** The lowest level of abstraction describes how the data are actually stored.



# Three layer architecture



As we can see from picture in the previous slide, there are three levels of schemas in the database architecture

**External level:**

- In this highest level, there exists a number of views which of is defined a part of the actual database.
- Each view is provided for a user or a group of users so that it helps in simplified the interaction between the user and system.

**Conceptual level:** Conceptual Schema in this level describes the logical structure of the whole database.

- The entire database is described using simple logical concepts such as objects, their properties or relationships. Thus the complexity of the implementation detail of the data will be hidden from the users.

**Internal level:** Internal Schema in this level describes how the data are actually stored, how to access the data.

## Internal level....

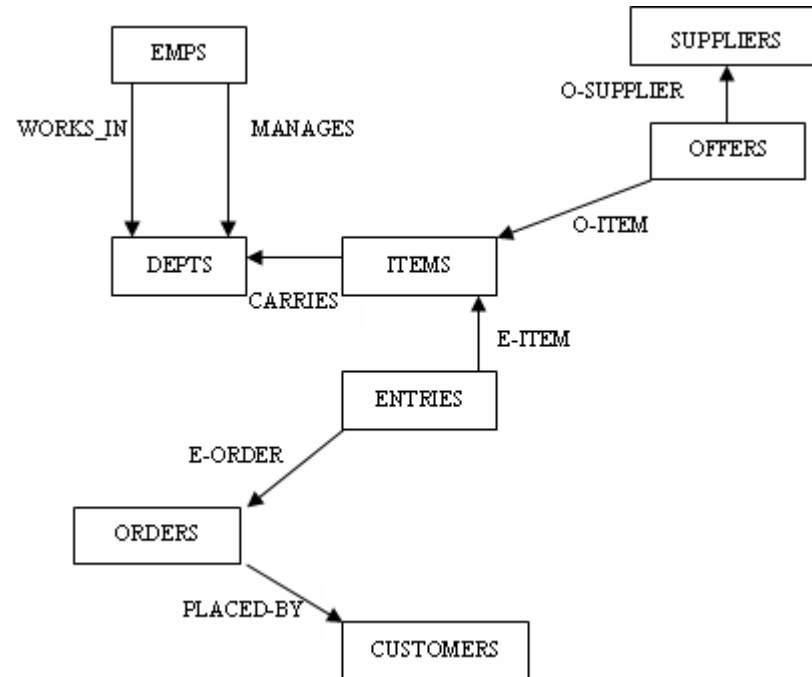
The three best-known models of this kind are the **relational data model**, the **network data** model and the **hierarchical data** model. These internal models:

- Consider a database as a collection of fixed-size records
- Are closer to the physical level or file structure
- Are a representation of the database as seen by the DBMS.
- Require the designer to match the conceptual model's characteristics and constraints to those of the selected implementation model
- Involve mapping the entities in the conceptual model to the tables in the relational model

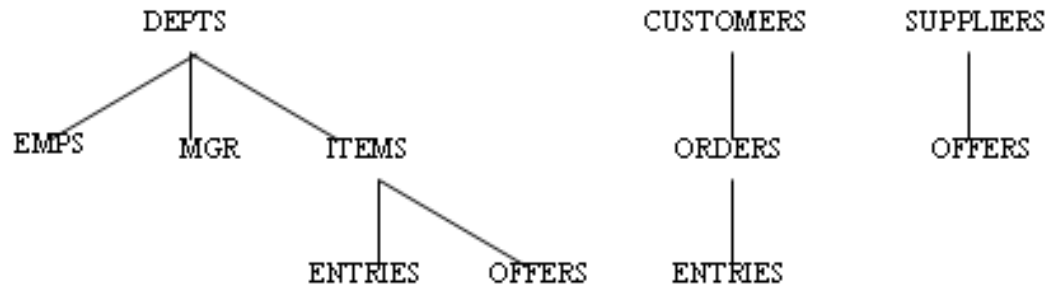
## Example of a **relational** schema for the SUPERMARKET database

EMPS (ENAME, SALARY)	MANAGERS (ENAME)
DEPTS (DNAME, DEPT#)	SUPPLIERS (SNAME, SADDR)
ITEMS (INAME, ITEM#)	ORDERS (O#, DATE)
WORKS_IN (ENAME, DNAME)	MANAGES (ENAME, DNAME)
CARRIES (INAME, DNAME)	PLACED_BY (O#, CNAME)
CUSTOMERS(CNAME,CADDR,BALANCE)SUPPLIES (SNAME, INAME, PRICES)INCLUDES (O#, INAME, QUANTITY)	

The figure below shows a schema for SUPERMARKET in **network model** notation



The figure below shows SUPERMARKET in a **hierarchical tree** structure



## Physical models

- Are the physical representation of the database
- Have the lowest level of abstractions
- Are how the data is stored; they deal with
  - Run-time performance
  - Storage utilization and compression
  - File organization and access methods
  - Data encryption
- Are the physical level – managed by the *operating system* (OS)
- Provide concepts that describe the details of how data are stored in the computer's memory



## Data Independence

Data Independence is the ability to modify the schema in one level without affecting the schema in the higher level.

There are two levels of data independence:

- **Logical data independence** is the ability to make change in the conceptual schema without causing a modify in the user views or application program.
- **Physical data independence** is the ability to make change in the internal schema without causing a modify in the conceptual schema or application program.

# Classification of Database Management Systems

Session 6

# Session 6

- Classification Based on Data Model
- Classification Based on User Numbers
- Classification Based on Database Distribution
- Distributed database system



## Database Language

- **Data Definition Language (DDL)**: This is used to define the conceptual and internal schemas for a database system.
  - It is not procedural language, rather a language for describing the types of entities and relationships among them in terms of a particular data model.
- **Data Manipulation Language (DML)**: This is used to manipulate the database, which typically include retrieval, insertion, deletion, and modification of the data.

## Classification of Database Systems

The database management systems can be classified based on several criteria.

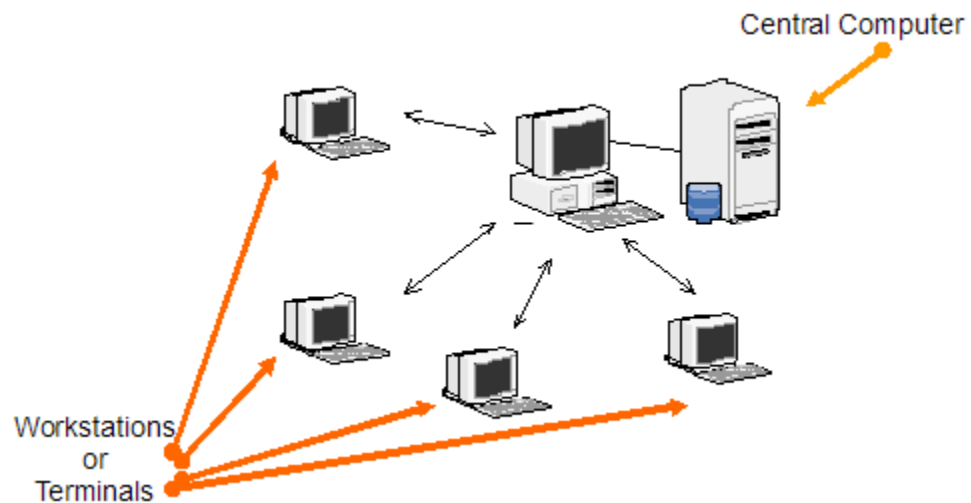
- **Based on data model:** The most popular data model in today commercial DBMSs is relational data model. Almost well-known DBMSs like Oracle, MS SQL Server, DB2, MySQL are support this model. Other traditional models can be named hierarchical data model , network data model. Also, we are getting familiar with object-oriented data model. Some examples of Object-oriented DBMSs are O2, ObjectStore or Jasmine.

## Classification of Database Systems

The database management systems can be classified based on several criteria.

- **Based on number users** we can have single user database system which support one user at a time or multiuser systems which support multiple users concurrently
- **Based on the ways database is distributed** we have centralized or distributed database system
  - **Centralized database system:** Data in this kind of system is stored at a single site.

## Centralized database system



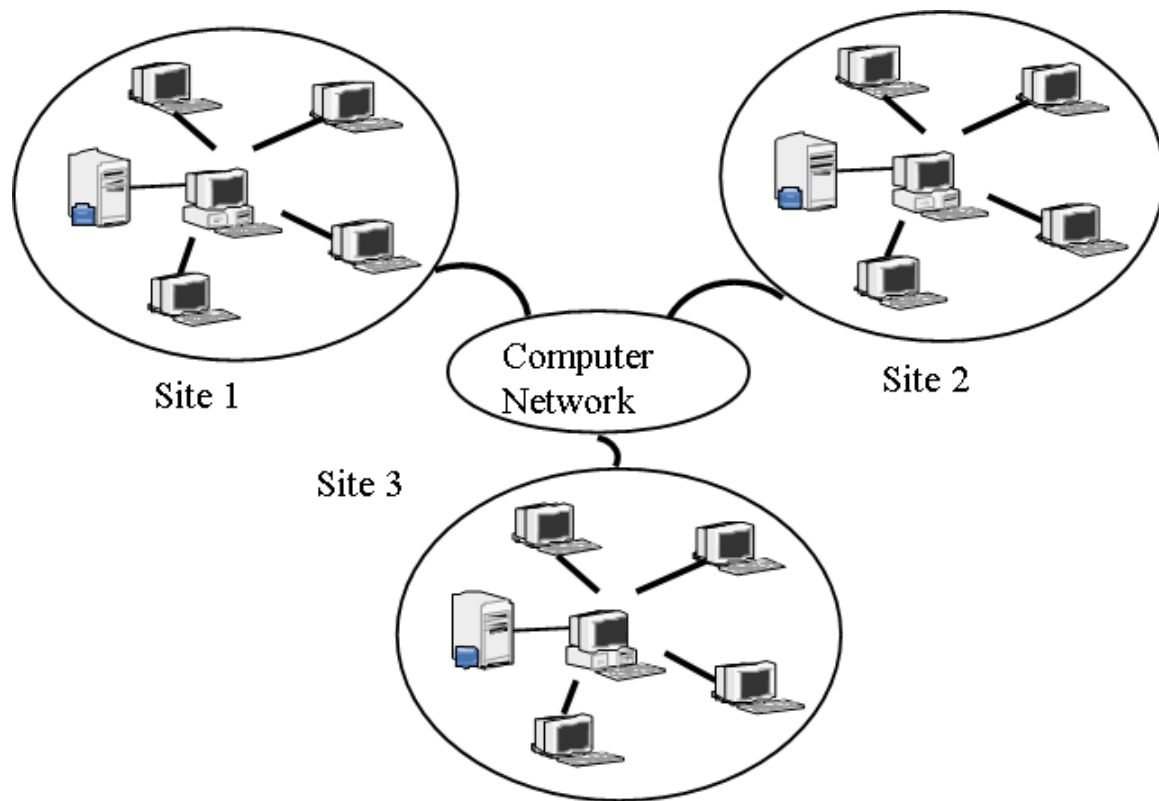
## Classification of Distributed Database Systems

**Distributed database system:** Actual database and DBMS software are distributed in various sites connected by a computer network.

- Homogeneous distributed Database Systems
  - Use the same DBMS software at multiple sites
  - Data exchange between various sites can be handle easily
- Heterogeneous distributed Database Systems
  - Different sites might use different DBMS software
  - There is a software to support data exchange between sites



## Distributed database system



# The Relational Data Model

Session 7

# Session 7

- Fundamental Concepts in the Relational Data Model
- Properties of a Table



## Relation

A *relation*, also known as a **table** or *file*, is a subset of the Cartesian product of a list of domains characterized by a name. And within a table, each row represents a group of related data values.

A **row**, or record, is also known as a *tuple*.

The **columns** in a table is a field and is also referred to as an attribute.

You can also think of it this way: an attribute is used to define the record and a record contains a set of attributes.

## Properties of a Table

- A table has a name that is distinct from all other tables in the database.
- There are no duplicate rows; each row is distinct.
- Entries in columns are atomic. The table does not contain repeating groups or multivalued attributes.
- Entries from columns are from the same domain based on their data type including:
  - number (numeric, integer, float, smallint,...)
  - character (string)
  - date
  - logical (true or false)

## Properties of a Table

- Operations combining different data types are disallowed.
- Each attribute has a distinct name.
- The sequence of columns is insignificant.
- The sequence of rows is insignificant.

## Key Terms

- **atomic value**: each value in the domain is indivisible as far as the relational model is concerned
- **attribute**: principle storage unit in a database
- **column**: see *attribute*
- **degree**: number of attributes in a table
- **domain**: the original sets of atomic values used to model data; a set of acceptable values that a column is allowed to contain
- **field**: see *attribute*
- **record**: contains fields that are related;

# The Entity Relationship Data Model

Session 8



# Session 8

- Entity, Entity Set and Entity Type
- Kinds of Entities
- Attributes & Types of Attributes
- Keys & Types of Keys
- Nulls
- Relationships & Types of Relationships



## ER modeling is based on two concepts:

- Entities, defined as tables that hold specific information (data)
- *Relationships*, defined as the associations or interactions between entities

## Entity, Entity Set and Entity Type

An **entity** is an object in the real world with an independent existence that can be differentiated from other objects. An entity might be

- An object with physical existence (e.g., a lecturer, a student, a car)
- An object with conceptual existence (e.g., a course, a job, a position)

## Entity, Entity Set and Entity Type

Entities can be classified based on their strength. An entity is considered weak if its tables are existence dependent.

- That is, it cannot exist without a relationship with another entity
- Its primary key is derived from the primary key of the parent entity
  - The Spouse table, in the COMPANY database, is a weak entity because its primary key is dependent on the Employee table. Without a corresponding employee record, the spouse record would not exist.

## Entity, Entity Set and Entity Type

An entity is considered strong if it can exist apart from all of its related entities.

- Kernels are strong entities.
- A table without a foreign key or a table that contains a foreign key that can contain nulls is a strong entity
- Another term to know is *entity type* which defines a collection of similar entities.

## Entity, Entity Set and Entity Type

Another term to know is *entity type* which defines a collection of similar entities.

An *entity set* is a collection of entities of an entity type at a particular point of time. In an entity relationship diagram (ERD), an entity type is represented by a name in a box

## Kinds of Entities

You should also be familiar with different kinds of entities including **independent** entities, **dependent** entities and **characteristic** entities.

## Independent entities

*Independent entities*, also referred to as kernels, are the backbone of the database. They are what other tables are based on. *Kernels* have the following characteristics:

- They are the building blocks of a database.
- The primary key may be simple or composite.
- The primary key is not a foreign key.
- They do not depend on another entity for their existence.



## Dependent entities

*Dependent entities*, also referred to as *derived entities*, depend on other tables for their meaning. These entities have the following characteristics:

- Dependent entities are used to connect two kernels together.
- They are said to be existence dependent on two or more tables.
- Many to many relationships become associative tables with at least two foreign keys.
- They may contain other attributes.
- The foreign key identifies each associated table.
- There are three options for the primary key:
  - Use a composite of foreign keys of associated tables if unique
  - Use a composite of foreign keys and a qualifying column
  - Create a new simple primary key

## Characteristic entities

- *Characteristic entities* provide more information about another table. These entities have the following characteristics:
  - They represent multi-valued attributes.
  - They describe other entities.
  - They typically have a one to many relationship.
  - The foreign key is used to further identify the characterized table.
  - Options for primary key are as follows:
    - Use a composite of foreign key plus a qualifying column
    - Create a new simple primary key. In the COMPANY database, these might include:
      - Employee (EID, Name, Address, Age, Salary) – EID is the simple primary key.
      - EmployeePhone (EID, Phone) – EID is part of a composite primary key. Here, EID is also a foreign key.

## Types of Attributes

- Simple attributes
- Composite attributes
- Multivalued attributes
- Derived attributes

## Types of Keys

- Candidate key
- Composite key
- Primary key
- Alternate key
- Foreign key

## Nulls

- A *null* is a special symbol, independent of data type, which means either unknown or inapplicable. It does not mean zero or blank. Features of null include:
  - No data entry
  - Not permitted in the primary key
  - Should be avoided in other attributes
  - Can represent
    - An unknown attribute value
    - A known, but missing, attribute value
    - A “not applicable” condition
- Can create problems when functions such as COUNT, AVERAGE and SUM are used
- Can create logical problems when relational tables are linked

## Nulls- Examples

```
SELECT emp# FROM Salary_tbl  
WHERE jobName = Sales AND  
(commission + salary) > 30,000 → E10 and E12
```

```
SELECT emp# FROM Salary_tbl  
WHERE jobName = Sales AND  
(commission > 30000 OR  
salary > 30000 OR  
(commission + salary) > 30,000 → E10 and E12 and E13
```

## Types of relationship

- One to many (1:M) relationship
- One to one (1:1) relationship
- Many to many (M:N) relationships
- Unary relationship (recursive)
- Ternary Relationships

# Integrity Rules and Constraints

Session 9



# Session 9

- Domain Integrity
- Enterprise Constraints & Business Rules
- Relationship Types



**Constraints** are a very important feature in a relational model. In fact, the relational model supports the well-defined theory of constraints on attributes or tables. Constraints are useful because they allow a designer to specify the semantics of data in the database.

*Constraints* are the rules that force DBMSs to check that data satisfies the semantics.

There are several kinds of integrity constraints, described below:

### Entity integrity

To ensure *entity integrity*, it is required that every table have a primary key.

### Referential integrity

*Referential integrity* requires that a foreign key must have a matching primary key or it must be null. This constraint is specified between two tables (parent and child); it maintains the correspondence between rows in these tables. It means the reference from a row in one table to another table must be valid.

## Enterprise Constraints

Enterprise constraints – sometimes referred to as semantic constraints – are additional rules specified by users or database administrators and can be based on multiple tables.

Here are some examples.

- A class can have a maximum of 30 students.
- A teacher can teach a maximum of four classes per semester.
- An employee cannot take part in more than five projects.
- The salary of an employee cannot exceed the salary of the employee's manager.

## Business Rules

*Business rules* are obtained from users when gathering requirements. The requirements-gathering process is very important, and its results should be verified by the user before the database design is built. If the business rules are incorrect, the design will be incorrect, and ultimately the application built will not function as expected by the users.

Some examples of business rules are:

- A teacher can teach many students.
- A class can have a maximum of 35 students.
- A course can be taught many times, but by only one instructor.
- Not all teachers teach classes.

## Relationship Types

- Optional relationships
- Mandatory relationships

# ER Modeling

Session 10

# Session 10

- Relational Design and Redundancy
- Insertion Anomaly
- Update Anomaly
- Deletion Anomaly
- How to Avoid Anomalies





## Business Rules

One important theory developed for the entity relational (ER) model involves the notion of **functional dependency** (FD).

Like constraints, FDs are drawn from the semantics of the application domain. Essentially, *functional dependencies* describe how individual attributes are related. FDs are a kind of constraint among attributes within a relation and contribute to a good relational schema design.

In this session, we will look at:

- The basic theory and definition of functional dependency
- The methodology for improving schema designs, also called normalization

## Relational Design and Redundancy

Generally, a good relational database design must capture all of the necessary attributes and associations. The design should do this with a minimal amount of stored information and no redundant data.

## Insertion Anomaly

An *insertion anomaly* occurs when you are inserting inconsistent information into a table.

## Update Anomaly

## Deletion Anomaly

A *deletion anomaly* occurs when you delete a record that may contain attributes that shouldn't be deleted.

## How to Avoid Anomalies

The best approach to creating tables without anomalies is to ensure that the tables are normalized, and that's accomplished by understanding functional dependencies. FD ensures that all attributes in a table belong to that table. In other words, it will eliminate redundancies and anomalies.

# Recap

What we have learned so far?

# Next Session

- Exercises and projects



# Private Coaching Sessions

Private tutoring sessions for system administrator  
management- Weekly and monthly plans

Database design and SQL coding- Private tutoring sessions

# Next Classes

- Intro to SQL programming
- Introduction to Linux OS
- Learn PHP Programming
- Web Development with PHP & MySQL
- Learn Node.JS, Express.JS  
and MongoDB





coding-bootcamps.com

Thank you



**Coding**  
Bootcamps