# RDBMS Fundamentals

## What is Database?

Database is a systematic collection of data. We can store and manipulate data. Data management is easy.

Examples:

* Your Contact application uses database to store name, numbers, email id and address.
* Facebook uses database to store images, videos, text, user activities, and many other things.

## Database Applications

* Banking: Customer’s data, Transaction history
* Airlines: Reservation and Schedules
* Universities: Registration and Grade
* Sales: Consumer’s data, products, purchases
* Online Retailers: Order tracking, customized recommendations
* Human Resources: Employee Record, salaries, and tax deduction

## History of Database Management System

* 1960 - Charles Bachman designed first DBMS system.
* 1970 - Codd introduced IBM'S Information Management System (IMS).
* 1976 - Peter Chen coined and defined the Entity-relationship model also known as the ER model.
* 1980 - Relational model becomes a widely accepted database component.
* 1985 - Object-oriented DBMS develops.
* 1990 - Incorporation of object-orientation in relational DBMS.
* 1991 - Microsoft ships MS access, a personal DBMS and that displaces all other personal DBMS products.
* 1995 - First Internet database applications.
* 1997 - XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

## Advantages of Database Management System:

* It offers variety of technique to store and retrieve data
* Data is secured
* Faster data access
* Data is consistent and accurate in the database
* Backup and Recovery

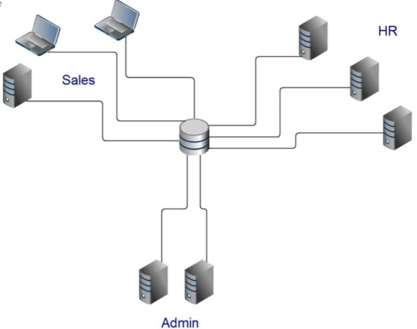
# Types of Databases

* Centralized Database ​
* Distributed Database​
* Relational Database​
* NoSQL Database​
* Document oriented database​
* Embedded database​
* Graph database​

## Centralized Database

* A single central database accessed by multiple users​
* Easier to organize, edit, query and backup​
* Can be slower because of high usage / load​
* Since all the data is in one place, there can be stronger security measures around it, so it is much more secure​
* It is cheaper than other databases as it requires less power and maintenance​

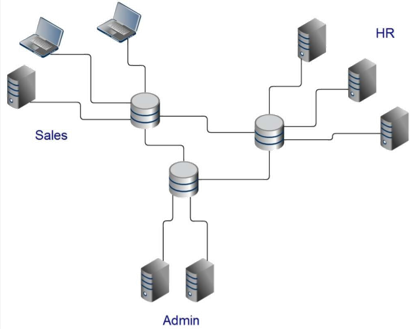
Example:



## Distributed Database​

* It is a database in which portions of the database are stored on multiple computers within a network​
* The DDBMS synchronizes all the data periodically​
* Updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere​
* Data access and retrieval faster at nearest points​

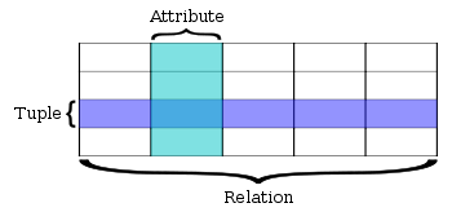
Example:



## Relational Database​

* A relational database is a database that has a collection of tables of data items​
* Data in a single table represents a relations​
* Tables consist of rows and columns​
* The columns constitute the attributes​
* The rows constitute tuples​
* A primary key is used to uniquely identify each row​
* The foreign key a column or group of columns in one table that points to the primary key of another table​

Example:



* **Row-A data set representing a single item​**
* **Column-A labeled element of a tuple​**

**e.g., "Address" or "Date of birth”, of multiple data item​**

## NoSQL Database​

* Stands for Not Only SQL​
* Term was redefined by Eric Evans after Carlo Strozzi​
* Class of non-relational data storage systems​
* Do not require a fixed table schema nor do they use the concept of joins​
* Relaxation for one or more of the ACID properties (Atomicity, Consistency, Isolation, Durability) using CAP theorem​

### Need of NoSQL

* Explosion of social media sites (Facebook, Twitter, Google etc.) with large data needs​
* Rise of cloud-based solutions such as Amazon S3 (simple storage solution)​
* Just as moving to dynamically typed languages (Ruby/Groovy), a shift to dynamically typed data with frequent schema changes​
* Expansion of Open source community​
* NoSQL solution is more acceptable to a client now than a year ago​

### Types of NoSQL

NoSQL databases are classified into four types:​

* **Key Value pair based:**Designed for processing dictionary. Dictionaries contain a collection of records having fields containing data​
* **Column based:**It store data as Column families containing rows that have many columns associated with a row key. Each row can have different columns​
* **Document based:**The database stores and retrieves documents. It stores documents in the value part of the key-value store​
* **Graph based:**Store entities and relationships between these entities as nodes and edges of a graph respectively. Entities have properties​

# ACID

* A transaction is a single logical unit of work which accesses and possibly modifies the contents of a database​
* Transactions access data using read and write operations​
* To maintain consistency in a database, before and after the transaction, certain properties are followed​
* These are called **ACID** properties​

​

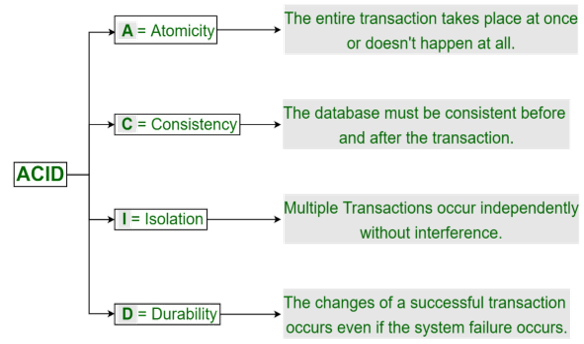
ACID in DBMS stands for ​

**A**tomicity​

**C**onsistency​

**I**solation​

**D**urability​

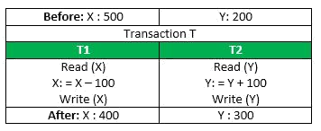


### Atomicity

* By this, we mean that either the entire transaction takes place at once or doesn’t happen at all​
* There is no midway i.e., transactions do not occur partially​
* Each transaction is considered as one unit and either runs to completion or is not executed at all​
* It involves the following two operations​
  + **Abort**: If a transaction aborts, changes made to database are not visible​
  + **Commit**: If a transaction commits, changes made are visible​
* Atomicity is also known as the ‘All or nothing rule’​

Example:

* Consider the following transaction **T** consisting of **T1** and **T2**: Transfer of 100 from account **X** to account **Y**​



* If the transaction fails after completion of **T1** but before completion of **T2**. (Say, after **write(X)** but before **write(Y)**), then amount has been deducted from **X** but not added to **Y**​
* This results in an inconsistent database state​
* Therefore, the transaction must be executed in entirety to ensure correctness of database state. ​

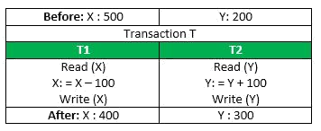
### Consistency

* A transaction must transform the database from one consistent state to another state​
* It is the responsibility of both the DBMS and the application developers to ensure consistency​
* This means that integrity constraints must be maintained so that the database is consistent before and after the transaction​
* It refers to the correctness of a database​

Example:

Referring to the previous example​

* The total amount before and after the transaction must be maintained​



​

Total **before T** occurs = **500 + 200 = 700**​  
Total **after T occurs** = **400 + 300 = 700**​  
​

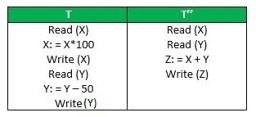
* Therefore, database is **consistent**​
* Inconsistency occurs in case **T1** completes but **T2** fails​
* As a result, T is incomplete​

### Isolation

* This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of database state​
* Transactions occur independently without interference​
* Changes occurring in a particular transaction will not be visible to any other transaction until that change in that transaction is written to memory or has been committed​
* This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order​

Example:

* Let **X**= 500, **Y** = 500. ​  
  Consider two transactions **T** and **T”**​



* Suppose **T** has been executed till **Read (Y)** and then **T’’** starts​
* As a result , interleaving of operations takes place due to which **T’’** reads correct value of **X** but incorrect value of **Y** and sum computed by ​  
  **T’’: (X+Y = 50, 000+500=50, 500)** ​  
  is thus not consistent with the sum at end of transaction: ​  
  **T: (X+Y = 50, 000 + 450 = 50, 450)**​  
  This results in database inconsistency, due to a loss of 50 units​
* Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory​

### Durability

* This property ensures that once the transaction has completed execution​
* The updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs​
* These updates now become permanent and are stored in non-volatile memory​
* The effects of the transaction, thus, are never lost​

## ER Diagram

ER diagram stands for Entity Relationship Diagram. ER diagram helps us to explain logical structure of database. Diagrams are created based on three basic concepts: Entities, Attribute and Relationship.

This diagram contains different symbols like rectangle to represent entities, oval for attribute and diamond for relationship.

### Entity

* An object that exists​
* Distinguishable from other objects​
* Could be concrete or abstract​

Examples: a book, an item, a student, a purchase order​

(a/an above indicates that we are referring to one of these)​

### Entity Set

* A set of similar entities​
* Need not be disjoint with other entity sets ​

e.g., supplier and customer may have common entities​

* Example: set of all books in a library​

set of all customers​

* Entity set also called entity type or entity class​
* Entity considered as an occurrence of entity type​
* We often use the words ‘entity’ to mean ‘entity-set’​
* Entity sets are named using singular common nouns:​

Book​

Student​

Course​

### Attribute

* An entity has a set of attributes​
* Attribute defines property of an entity​
* It is given a name​
* Attribute has value for each entity​
* Value may change over time​
* Same set of attributes are defined for entities in an entity set​
* Example: Entity set BOOK has the following attributes​

TITLE ISBN​

ACC-NO AUTHOR​

PUBLISHER YEAR​

PRICE​

* A particular book has value for each of the above attributes​

​

* An attribute may be multi-valued, i.e., it has more than one value for a given entity, e.g., a book may have many authors​
* An attribute which uniquely identifies entities of a set is called primary key attribute of that entity set​
* Composite attribute: date, address, etc.

### Domain​

* Gives set of permitted values for an attribute​
* All values may not be always present in database​
* May be defined by type: integer, string​
* Attributes are ‘roles’ played by domains​
* Domain ‘person name’ can be used for attribute ‘name’ for teacher and student entities​

Example:  A COLLEGE​

STUDENT:  rollno, name, hostel-no., date-of-birth​

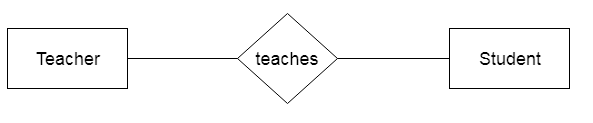
COURSE:  courseno, name, credits​

TEACHER:  empno, name, rank, room-no., tele-phone​

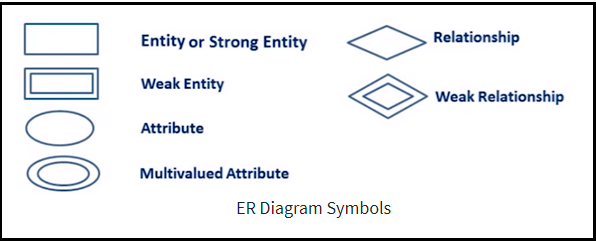
DEPT:  name, tele-phone​

### Relationship

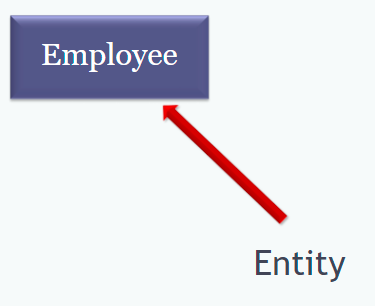
* Relationship between two or more entities
* It is represented by a diamond-shaped box
* Entities participating in relationship are connected using line



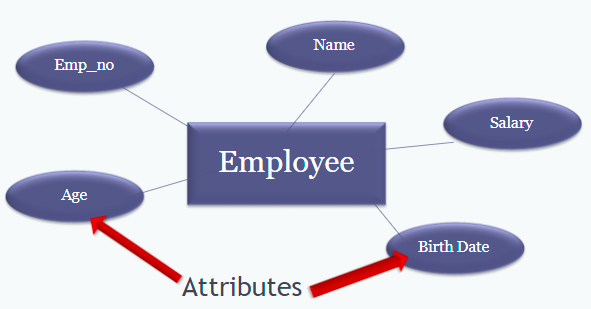
### ER Symbols



#### Entity

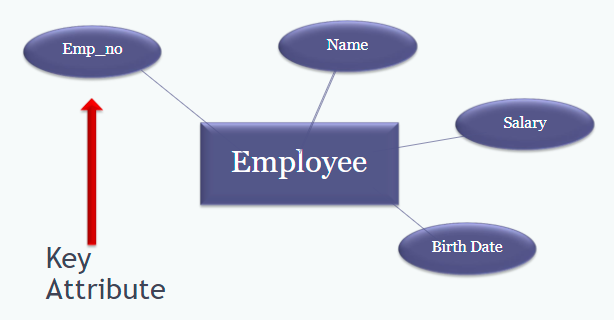


#### Attribute



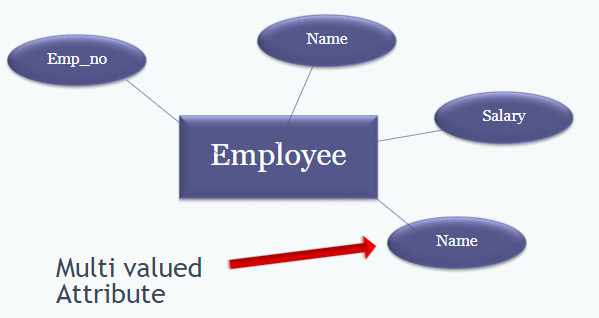
#### Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



#### Multi Values Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.



### Cardinality

It defines the numerical attribute of the relationship between two entities.

Types of Cardinalities:

* One-to-One Relationship
* One-to-Many Relationship
* Many-to-One Relationship
* Many-to-Many Relationship

#### One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one-to-one relationship.

Example: A single person will have only one passport.

#### One-to-Many Relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

Example: A customer can place many orders but a order cannot be placed by many customers.

#### Many-to-One Relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

Example: Many students can study in a single college, but a student cannot study in many colleges at a same time.

#### Many-to-Many Relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

Example: Students can be assigned to many projects and a project can be assigned to many students.

## Normalization

Normalization is the process of organizing the data in the database. It is used to minimize the redundancy from a relation or set of relations. Normal form is used to reduce redundancy from the database table. It divides the larger table into the smaller table and link them using relationship.

Goal of Normalization: producing well-structured relations by eliminating anomalies​.

### Types of Anomalies

* Update (Modification) Anomaly ​
  + Changing data in a row force changes to other rows because of duplication​
* Deletion Anomaly ​
  + Deleting rows may cause a loss of data that would be needed for other future rows​
* Insertion Anomaly ​
  + Adding new rows forces user to create duplicate data​

### Types of Normal Forms

* 1NF
* 2NF
* 3NF

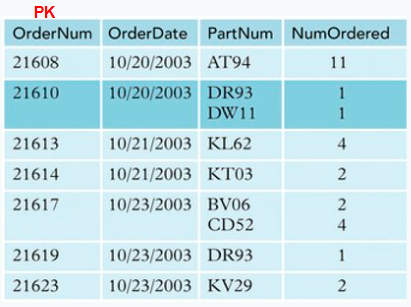
### First Normal Form(1NF)

To be in First Normal Form (1NF):

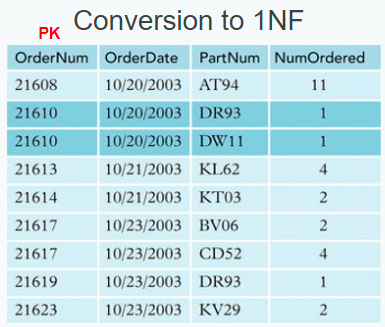
* + Each column must contain only a single value​
  + Repeating groups of records (redundancy) must be eliminated​
  + Eliminate duplicative columns from the same table​
  + There must not be a composite and a multi-valued attributes​
  + Transformation from model to relation​

Example:

Unnormalized Table-



After Conversion to 1NF-



### Second Normal Form(2NF)

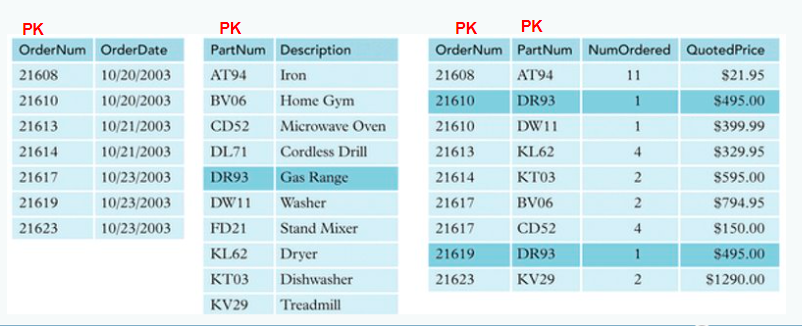
* To be in 2NF, a relation must be in 1NF, and a relation must **not have any partial dependencies**​
  + Any attributes must not be dependent on a portion of primary key​
* The other way to understand 2NF is that each non-key attribute (not a part of PK) in the relation must be functionally dependent upon the primary key ​

Example:

Unnormalized Table-



After Conversion to 2NF-



### Third Normal Form(3NF)

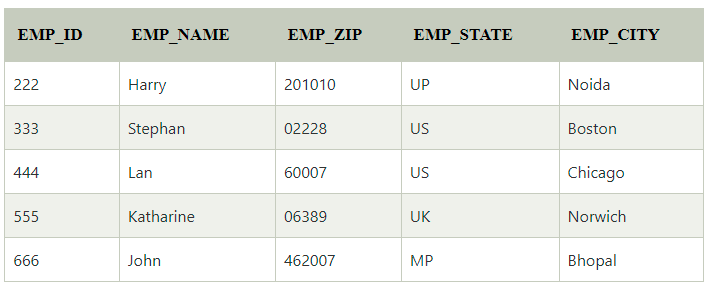
* To be in Third Normal Form, a relation must first fulfill the requirements to be in 2NF​
* Additionally, all attributes that are not dependent upon the primary key must be eliminated. In other words, there should be no transitive dependencies​
  + remove columns that are not dependent upon the primary key​
* 3NF is used to reduce the data duplication. It is also used to achieve the data integrity

A relation is in third normal form if it holds at least one of the following conditions for every non-trivial function dependency X → Y.

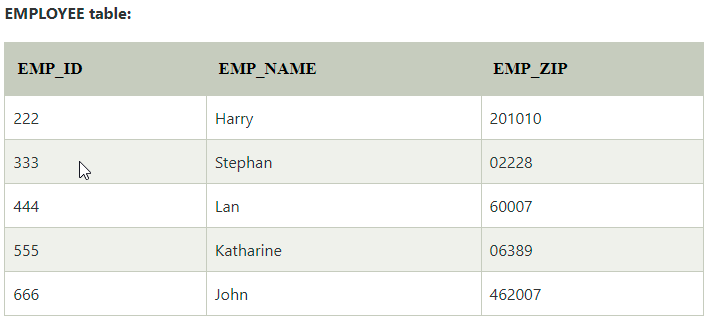
* X is a super key.
* Y is a prime attribute, i.e., each element of Y is part of some candidate key.

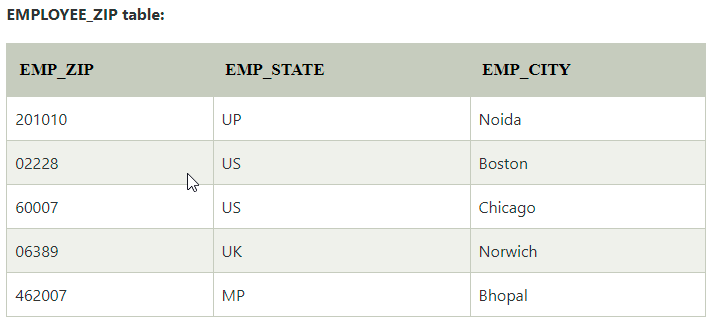
Example:

Unnormalized Table-



After Conversion to 3NF-





## Table and Keys

### Table

The data in an RDBMS is stored in database objects which are called as tables. This table is basically a collection of related data entries, and it consists of numerous columns and rows.

### Keys

Keys plays an important role in relational database. It is used to uniquely identify any record from the table. It is used to connect two tables.

#### Types of keys

* Primary key
* Candidate key
* Super key
* Foreign key
* Composite key
* Alternate key
* Unique key

##### Primary Key

It’s a unique key in entire table. Using this key, we can identify every record present in that table. There can be only one primary key. Every value of primary key must be unique with no repetitions.

Example: Employee\_Id, License\_number, Passport\_number

##### Candidate Key

A candidate key is an attribute or set of an attribute which can uniquely identify a tuple. The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key. The Primary Key of a table is selected from one of the candidate keys.

Example: In Employee table, Employee\_Id is the primary key and rest other attributes like Passport\_number, License\_number are Candidate key.

##### Super Key

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

Example: In the EMPLOYEE table, for (EMPLOEE\_ID, EMPLOYEE\_NAME) the name of two employees can be the same, but their EMPLYEE\_ID can't be the same. Hence, this combination can also be a key.

##### Foreign Key

Foreign Key is used to establish relationships between two tables. A foreign key will require each value in a column or set of columns to match the Primary Key of the referential table. Foreign keys help to maintain data and referential integrity.

Example: In a company, every employee works in a specific department, and employee and department are two different entities. So, we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.

We add the primary key of the DEPARTMENT table, Department\_Id as a new attribute in the EMPLOYEE table.

Now in the EMPLOYEE table, Department\_Id is the foreign key, and both the tables are related.

##### Composite Key

Key that consists of two or more attributes that uniquely identify any record in a table is called Composite key. But the attributes which together form the Composite key are not a key independently or individually.

Example: To store marks of student of each subject, we can use student\_id and subject\_id.

##### Alternate Key

The candidate key which are not selected as primary key are known as secondary keys or alternative keys.

##### Unique Key

Unique Key is a column or set of columns that uniquely identify each record in a table. All values will have to be unique in this Key. A unique Key differs from a primary key because it can have only one null value, whereas a primary Key cannot have any null values.