A day without new knowledge is a lost day.

Database Technologies - MySQL

If A and a, B and b, C and c etc. are treated in the same way then it is case-insensitive. **MySQL is case-insensitive**

In this module we are going to learn **SQL**, **PL/SQL** and **NoSQL(MongoDB)**

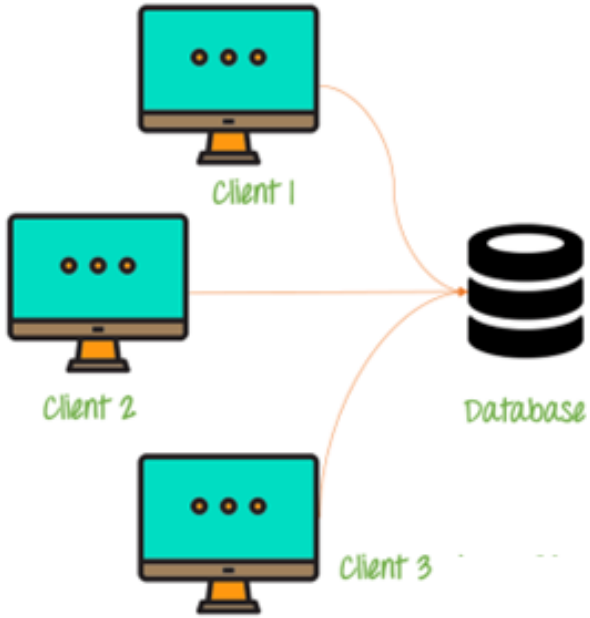
Introduction

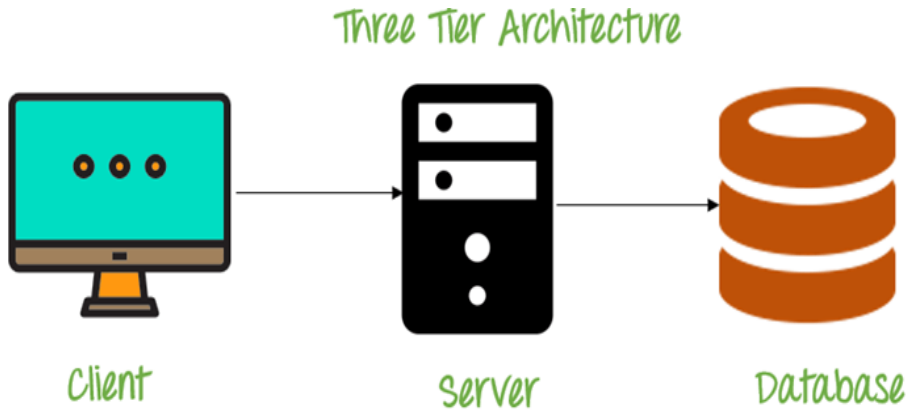
- If anyone who wants to develop a good application
  then he should have the knowledge three major components.

They are . . . . .

- Presentation Layer [UI]
- Application Layer [Server Application and Client Application]
- Data Layer [Data Access Object (DAO) / Data Access Layer (DAL)] { Flat Files | RDBMS | NoSQL }

Single Tier Architecture





Introduction

Why do we need databases (Use Case)?

We **need databases** because they organize data in a manner which allows us to store, query, sort, and manipulate data in various ways. Databases allow us to do all this things.

Many companies collects data from different resource (like Weather data, Geographical data, Finance data, Scientific data, Transport data, Cultural data, etc.)

What is Relation and Relationship?

Remember:

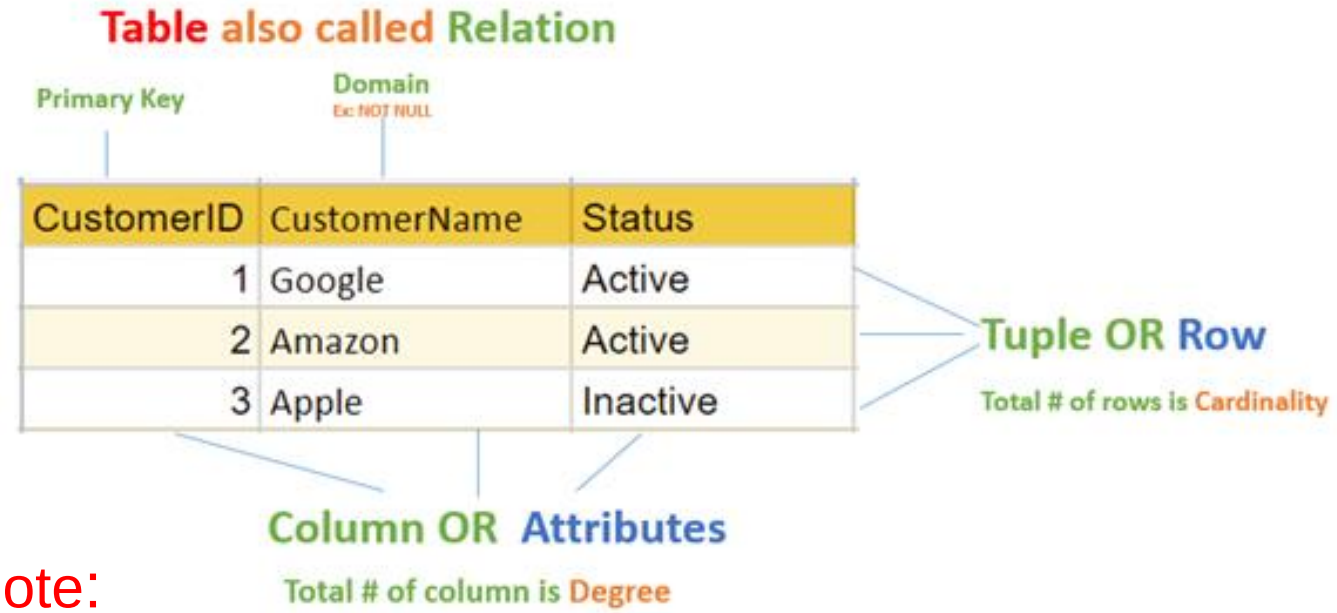
- A ***reference*** is a relationship between two tables where the values in one table refer to the values in another table.
- A ***referential key*** is a column or set of columns in a table that refers to the primary key of another table. It establishes a relationship between two tables, where one table is called the parent table, and the other is called the child table.

relation and relationship?

Relation ***(in Relational Algebra "R" stands for relation)*****:** In Database, a relation represents a **table** or an **entity** than contain attributes.

Relationship: In database, relationship is that how the two entities are **connected** to each other, i.e. what kind of relationship type they hold between them.

Primary/Foreign key is used to specify this relationship.



Remember:

Foreign Key is also know as
- referential constraint
- referential integrity constraint.

Note:

- **Table -** The physical instantiation of a relation in the database schema.
- **Relation -** A logical construct that organizes data into rows and columns.

File Systems is the traditional way to keep your data organized.

File System
vs
DBMS

file-oriented system

File Anomalies

```
struct Employee {
   int emp_no;
   char emp_name[50];
   int salary;
} emp[1000];
```

```
struct Employee {
   int emp_no;
   char emp_name[50];
   int salary;
};
struct Employee emp[1000];
```

c:\employee.txt

```
1   suraj 4000
2   ramesh 6000
3   rajan 4500
.       |
.       |
.       |
500 sam 3500
.       |
.       |
.       |
1000 amit 2300
```

c:\employee.txt

```
1   suraj 4000
2   ramesh 6000
3   rajan 4500
.       |
.       |
.       |
500 sam 3500
.       |
.       |
.       |
1000 amit 2300
.       |
.       |
.       |
2000 jerry 4500
.       |
.       |
```

c:\employee.txt

```
1   suraj 4000
2   ramesh 6000
3   rajan 4500
.       |
.       |
500  sam 3500
.       |
3   rajan 4500
.       |
500  sam 3500
.       |
.       |
1000 amit 2300
```

c:\employee.txt

```
1   suraj 4000
2   ramesh 6000
3   rajan 4500
.       |
.       |
sam 500 3500
.       |
ram  550 5000
.       |
1000 amit 2300
```

c:\employee.txt

```
1   suraj 4000
2   ramesh 6000
3   rajan 4500
.       |
500 sam 3500
.       |
600 neel 4500
```

- Create/Open an existing file
- Reading from file
- Writing to a file
- Closing a file

file-oriented system

File Anomalies

c:\employee.txt

1 suraj 4000
2 ramesh 6000
3 rajan 4500
.
.
.
500 sam 3500
.
.
.
1000 amit 2300

file attributes

- File Name
- Type
- Location

file permissions

- File permissions
- Share permissions

search empl ID=1

1 suraj 4000
2 ramesh 6000
3 rajan 4500
.
.
.
500 sam 3500
.
.
.
1000 amit 2300

search emp_name

1 suraj 4000
2 ramesh 6000
3 rajan 4500
.
.
.
500 sam 3500
.
.
.
1000 amit 2300

advantages of file-oriented system

The biggest advantage of file-based storage is that anyone can understand the system.

Advantage of File-oriented system

- **Backup**: It is possible to take faster and automatic back-up of database stored in files of computer-based systems.
- **Data retrieval:** It is possible to retrieve data stored in files in easy and efficient way.
- **Editing**: It is easy to edit any information stored in computers in form of files.
- **Remote access**: It is possible to access data remote location.
- **Sharing**: The files stored in systems can be shared among multiple users at a same time.

disadvantage of file-oriented system

The biggest disadvantage of file-based storage is as follows.

Disadvantage of File-oriented system

- **Data redundancy**: It is possible that the same information may be duplicated in different files. This leads to data redundancy results in memory wastage.

(Suppose a customer having both kind of accounts- saving and current account. In such a situation a customers detail are stored in both the file, saving.txt- file and current.txt- file , which leads to Data Redundancy.)

- **Data inconsistency**: Because of data redundancy, it is possible that data may not be in consistent state.

(Suppose customer changed his/her address. There might be a possibility that address is changed in only one file (saving.txt) and other (current.txt) remain unchanged.)

- **Limited data sharing**: Data are scattered in various files and also different files may have different formats (for example: .txt, .csv, .tsv and .xml) and these files may be stored in different folders so, due to this it is difficult to share data among different applications.
- **Data Isolation:** Because data are scattered in various files, and files may be in different formats (for example: .txt, .csv, .tsv and .xml), writing new application programs to retrieve the appropriate data is difficult.
- **Data security:** Data should be secured from unauthorized access, for example a account holder in a bank should not be able to see the account details of another account holder, such kind of security constraints are difficult to apply in file processing systems.

***Relation Schema*:** A relation schema represents name of the relation with its attributes.

- e.g. student (roll_no int, name varchar, address varchar, phone varchar and age int) is relation schema for STUDENT

DBMS

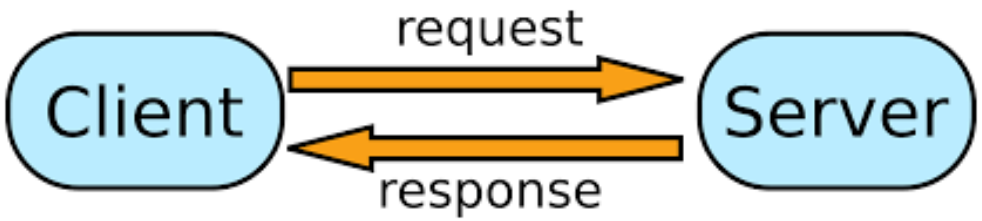
- **database:** Is the collection of **related data** which is **organized,** database can store and retrieve large amount of data easily, which is stored in one or more data files by one or more users, it is called as **structured data.**
- **management system**: it is a software, designed to **define**, **manipulate**, **retrieve** and **manage** data in a database.

relational database management system?

A RDBMS is a database management system (DBMS) that is based on the **relational model** introduced by Edgar Frank Codd at IBM in 1970.

RDBMS supports

- *client/server Technology*
- *Highly Secured*
- *Relationship (PK/FK)*



relational model concepts
and
properties of relational table

relational model concepts

Relational model organizes data into one or more tables (or "relations") of columns and rows. Rows are also called records or tuples. Columns are also called attributes.

- **Tables** − In relational model, relations are saved in the form of Tables. A table has rows and columns.
- **Attribute** − Attributes are the properties that define a relation. **e.g.** (**roll_no, name, address, phone and age)**
- **Tuple** − A single row of a table, which contains a single record for that relation is called a tuple.
- **Relation schema** − A relation schema describes the relation name (table name) with its attribute (columns) names.

  e.g. student(prn, name, address, phone, DoB, age, hobby, email, status) is relation schema for student relation.
- **Attribute domain** − An attribute domain specifies the data type, format, and constraints of a column, and defines the range of values that are valid for that column.

Remember:

- In database management systems, null is used to represent missing or unknown data in a table column.

properties of relational table

ID	job	firstName	DoB	salary
1	manager	Saleel Bagde	yyyy-mm-dd	••••••
3	salesman	Sharmin	yyyy-mm-dd	••••••
4	accountant	Vrushali	yyyy-mm-dd	••••••
2	salesman	Ruhan	yyyy-mm-dd	••••••
5	9500	manager	yyyy-mm-dd	••••••
5	Salesman	Rahul Patil	yyyy-mm-dd	••••••

Relational tables have six properties:

- Values are atomic.
- Column values are of the same kind. (***Attribute Domain*: Every attribute has some pre-defined datatypes, format, and constraints of a column, and defines the range of values that are valid for that column known as attribute domain.)**
- Each row is unique.
- The sequence of columns is insignificant – (unimportant).
- The sequence of rows is insignificant – (unimportant).
- Each attribute/column must have a unique name.

What is data?

what is data?

Data is any facts that can be stored and that can be processed by a computer.

Data can be in the form of Text or Multimedia

e.g.

- number, characters, or symbol
- images, audio, video, or signal

What is Entity Relationship Diagram?

Entity Relationship Diagram (ER Diagram)

Use E-R model to get a high-level graphical view to describe the **"ENTITIES"** and their **"RELATIONSHIP"**

The basic constructs/components of ER Model are **Entity**, **Attributes** and **Relationships**.

An entity can be a **real-world object.**

What is Entity?

entity

In relation to a database , an entity is a

- Person(student, teacher, employee, department, …)
- Place(classroom, building, …) --a particular position or area
- Thing(computer, lab equipment, …) --an object that is not named
- Concept(course, batch, student's attendance, …) -- an idea,

about which data can be stored. All these entities have some **attributes** or **properties** that give them their **identity**.

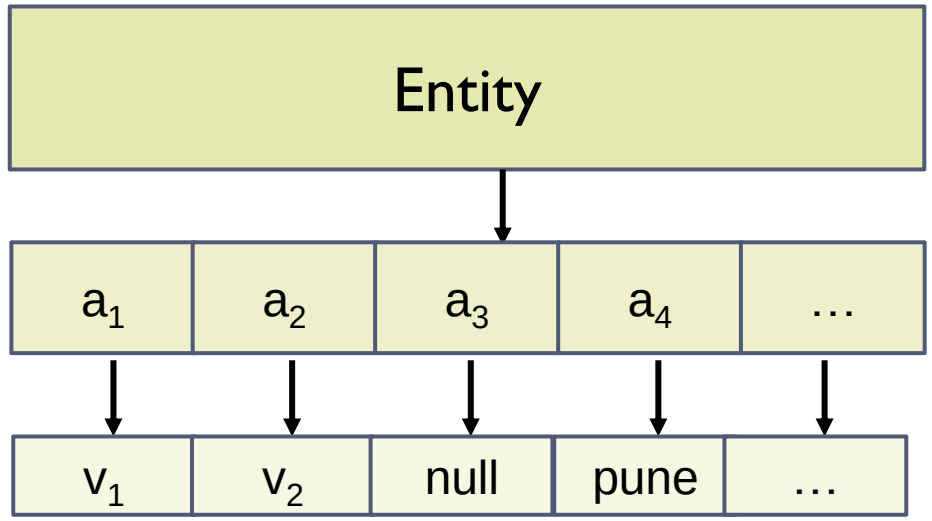
Every entity has its own characteristics.

In database management systems, **null** is used to represent missing or unknown data in a table column.

What is an Attribute?

Attributes are the properties that define a relation.

e.g. student(ID, firstName, middleName, lastName, city)



In Entity Relationship(ER) Model attributes can be classified into the following types.

- Simple/Atomic and Composite Attribute
- Single Valued and Multi Valued attribute
- Stored and Derived Attributes
- Complex Attribute

Remember:

In SQL, the same name can be used for two (or more) attributes as long as the attributes are in different relations.

attributes

- **Simple / Atomic Attribute** --VS-- **Composite Attribute**
  (Can't be divided further) (Can be divided further)
- **Single Value Attribute** --VS-- **Multi Valued Attribute**
  (Only One value) (Multiple values)
- **Stored Attribute** --VS-- **Derived Attribute**
  (Only One value) (Virtual)
- **Complex Attribute**
  (Composite & Multivalued)

Employee ID: An employee ID can be a composite attribute, which is composed of sub-attributes such as department code, job code, and employee number.

- **Atomic Attribute:** An attribute that cannot be divided into smaller independent attribute is known as atomic attribute.
  e.g. ID's, PRN, age, gender, zip, marital status cannot further divide.

- **Single Value Attribute:** An attribute that has only single value is known as single valued attribute.
  e.g. manufactured part can have only one serial number, voter card, blood group, price, quantity, branch can have only one value.

- **Stored Attribute:** The stored attribute are such attributes which are already stored in the database and from which the value of another attribute is derived.
  e.g. (HRA, DA…) can be derive from salary, age can be derived from DoB, total marks or average marks of a student can be derived from marks.

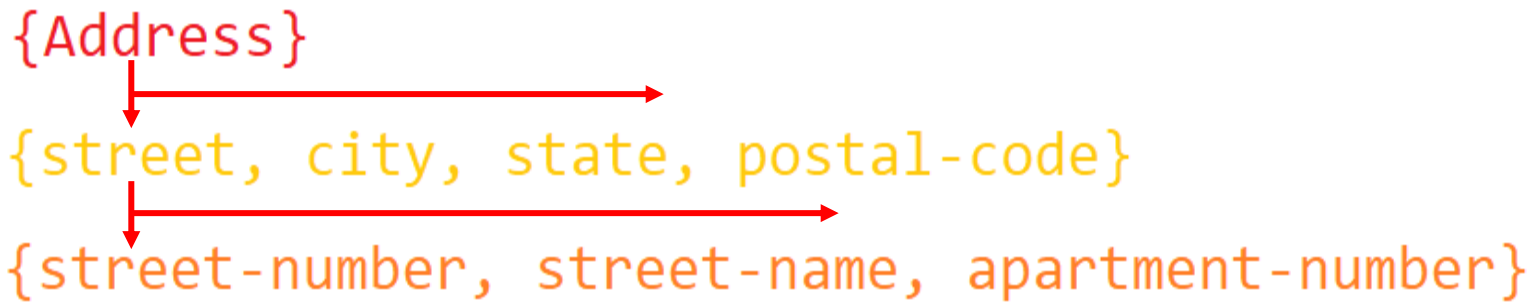
Composite **VS** Multi Valued Attribute

composite / multi valued attributes

Composite Attribute

Person Entity

- *Name* attribute: (firstName, middleName, and lastName)
- *PhoneNumber* attribute: (countryCode, cityCode, and phoneNumber)



Multi Valued Attribute

Person Entity

- *Hobbies* attribute: [reading, hiking, hockey, skiing, photography, . . .]
- *SpokenLanguages* attribute: [Hindi, Marathi, Gujarati, English, . . .]
- *Degrees* attribute**:** [10^{th} , 12^{th}, BE, ME, PhD, . . .]
- *emailID* attribute**:** [saleel@gmail.com, salil@yahoomail.com, . . .]

What is an Prime, Non-Prime Attribute?

Prime attribute (*Entity integrity*)

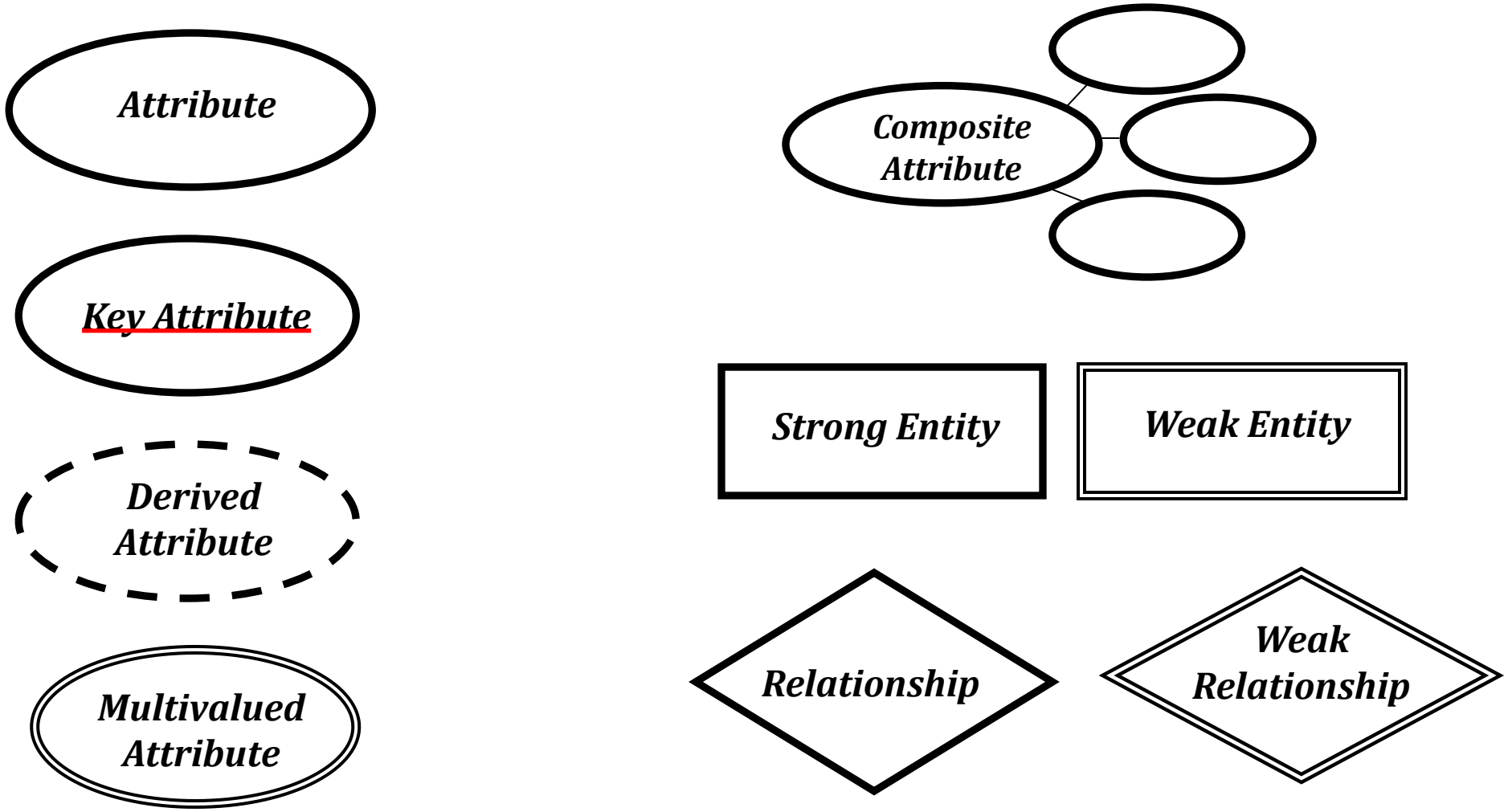
An attribute, which is a **part of the prime-key** (candidate key), is known as a prime attribute.

Non-prime attribute

An attribute, which is **not a part of the prime-key** (candidate key), is said to be a non-prime attribute.

Entity Relationship Diagram Symbols

entity relationship diagram symbols



strong and weak entity

An entity may participate in a relation either totally or partially.

Strong Entity: A strong entity is not dependent on any other entity in the schema. A strong entity will always have a **primary key**. Strong entities are represented by a single rectangle.

Weak Entity: A weak entity is dependent on a strong entity to ensure its existence. Unlike a strong entity, a weak entity does not have any primary key. A weak entity is represented by a double rectangle. The relation between one strong and one weak entity is represented by a double diamond.

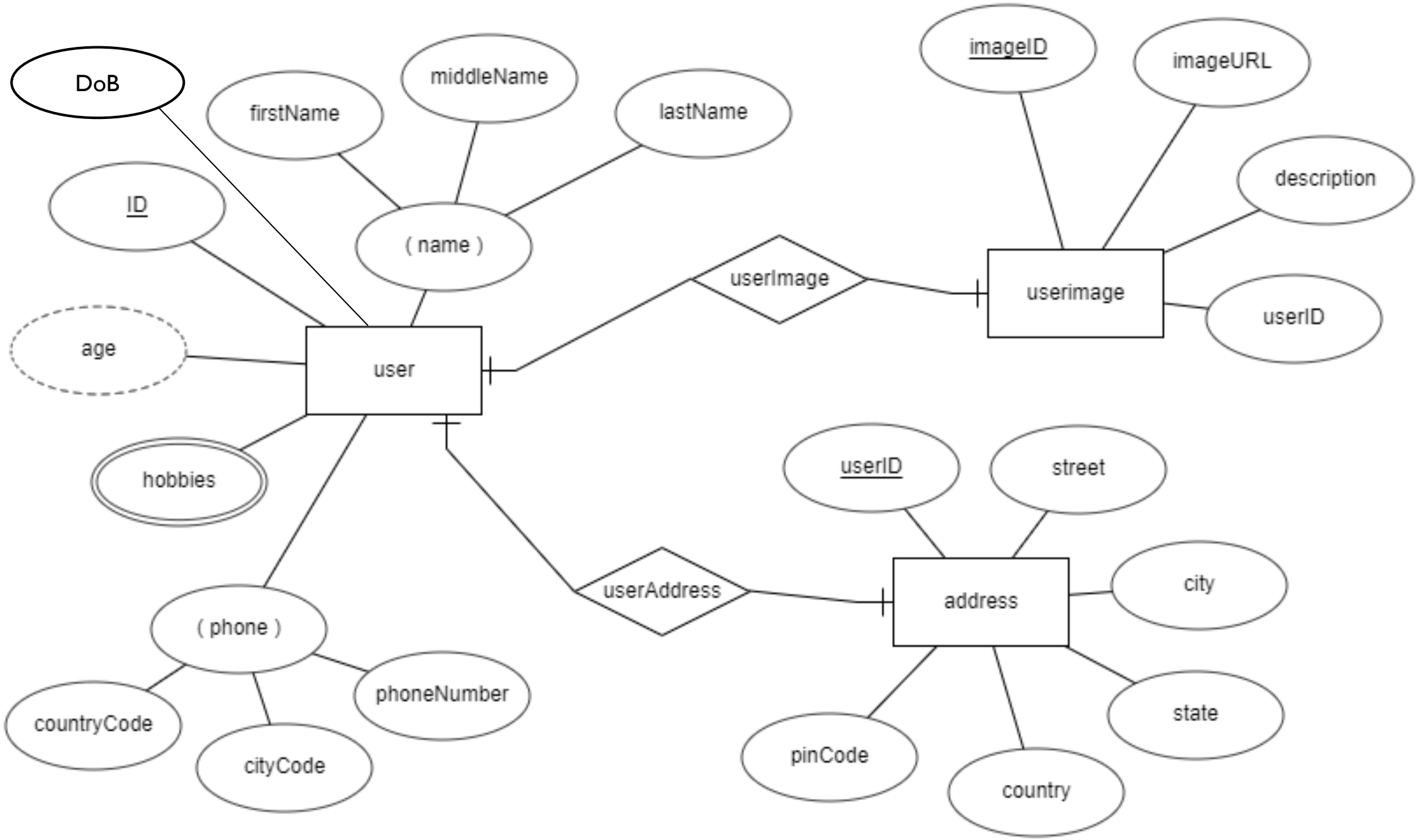
Example 1 – A loan entity can not be created for a customer if the customer doesn't exist

Example 2 – A payment entity can not be created for a loan if the loan doesn't exist

Example 3 – A dependents list entity can not be created if the employee doesn't exist

Example 4 – A prescription entity can not be created for a patient if the patient doesn't exist

entity relationship diagram



What is a degree, cardinality, domain and union in database?

What is a degree, cardinality, domain and union in database?

- **Degree d(R) / Arity**: Total number of **attributes/columns** present in a relation/table is called **degree of the relation** and is denoted by $\mathbf{d(}\boldsymbol{R}\mathbf{)}$.
- **Cardinality |R|**: Total number of **tuples/rows** present in a relation/table, **is called cardinality of a relation** and is denoted by $|\boldsymbol{R}|$.
- **Domain**: Total range of accepted values for an attribute of the relation **is called the domain of the attribute**. (**Data Type(size)**)
- **Union Compatibility**: Two relations ***R*** and ***S*** are set to be Union Compatible to each other if and only if:
  1. They have the **same degree $\mathbf{d(R)}$**.
  2. Domains of the respective attributes should also be same.

What is domain constraint and types of data integrity constraints?

Data integrity refers to the correctness and completeness of data.

A domain constraint and types of data integrity constraints

❖ **Domain Constraint** = data type + Constraints (**not null/unique/primary key/foreign key/check/default**)

e.g. custID INT, constraint pk_custid PRIMARY KEY(custID)

Three types of integrity constraints: **entity integrity, referential integrity** and **domain integrity**:

- **Entity integrity:** Entity Integrity Constraint is used to ensure the uniqueness of each record the table. There are primarily two types of integrity constraints that help us in ensuring the uniqueness of each row, namely, UNIQUE constraint and PRIMARY KEY constraint.

- **Referential integrity:** Referential Integrity Constraint ensures that there always exists a valid relationship between two tables. This makes sure that if a foreign key exists in a table relationship then it should always reference a corresponding value in the second table $t_1[\text{FK}] = t_2[\text{PK}]$ or it should be null.

- **Domain integrity:** A domain is a set of values of the same type. For example, we can specify if a particular column can hold null values or not, if the values have to be unique or not, the data type or size of values that can be entered in the column, the default values for the column, etc..