DBMS

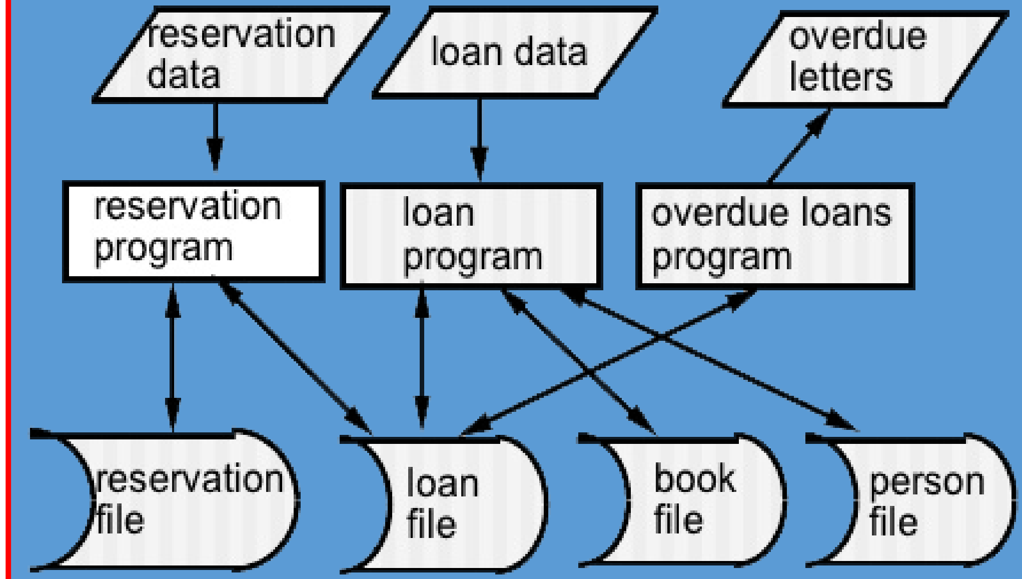
BASIC TERMINOLOGIES

- **Data** : Content that is entered into system
- **Field** : A character or group of characters (alphabetic or numeric) that has a specific meaning.
- **Record** : A logically connected set of one or more fields that describes a person, place, or thing.
- **File** : A collection of records.

FILE MANAGEMENT SYSTEM

- **File:** A collection of records or documents dealing with one organization, person, area or subject.

- Manual (paper) files
- Computer files



Files dedicated to application programs

DRAWBACKS OF FILE SYSTEMS

- **Data Inconsistency**
  - It Occurs When The Same data Doesn’t match each other In Various Relations(Tables).
- **Data redundancy**
  - It refers duplication of information in different files
- **Difficulty in accessing data**
  - Need to write a new program to carry out each new task

DBMS : INTRODUCTION

Database is a structured set of data held in a computer, especially one that is accessible in various ways.

A database management system (DBMS) is a collection of programs that enables you to store, modify, and extract information from a database.

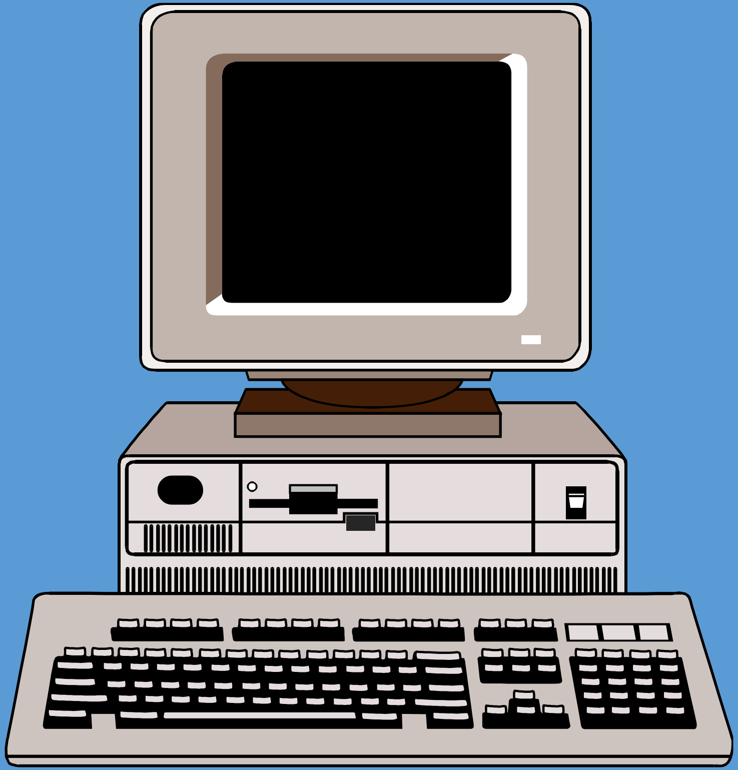
TYPES OF DATABASE SYSTEMS

- PC databases
- Centralized database
- Distributed databases

PC DATABASE

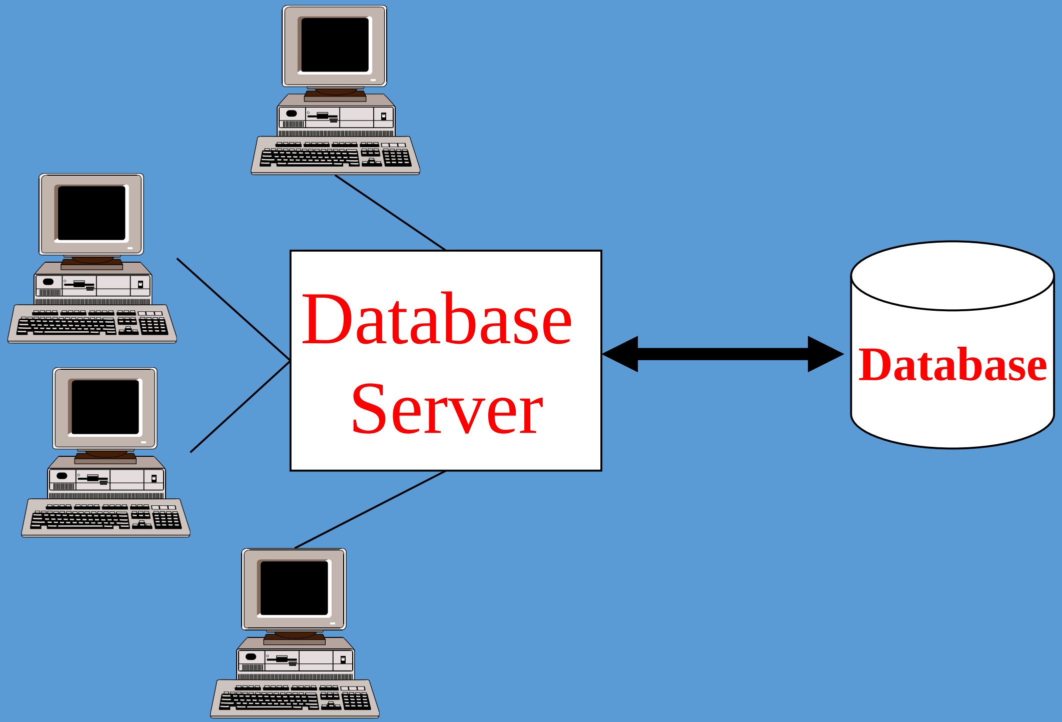
This type of data base is used for single PC.

Eg- MS-Access, foxpro.



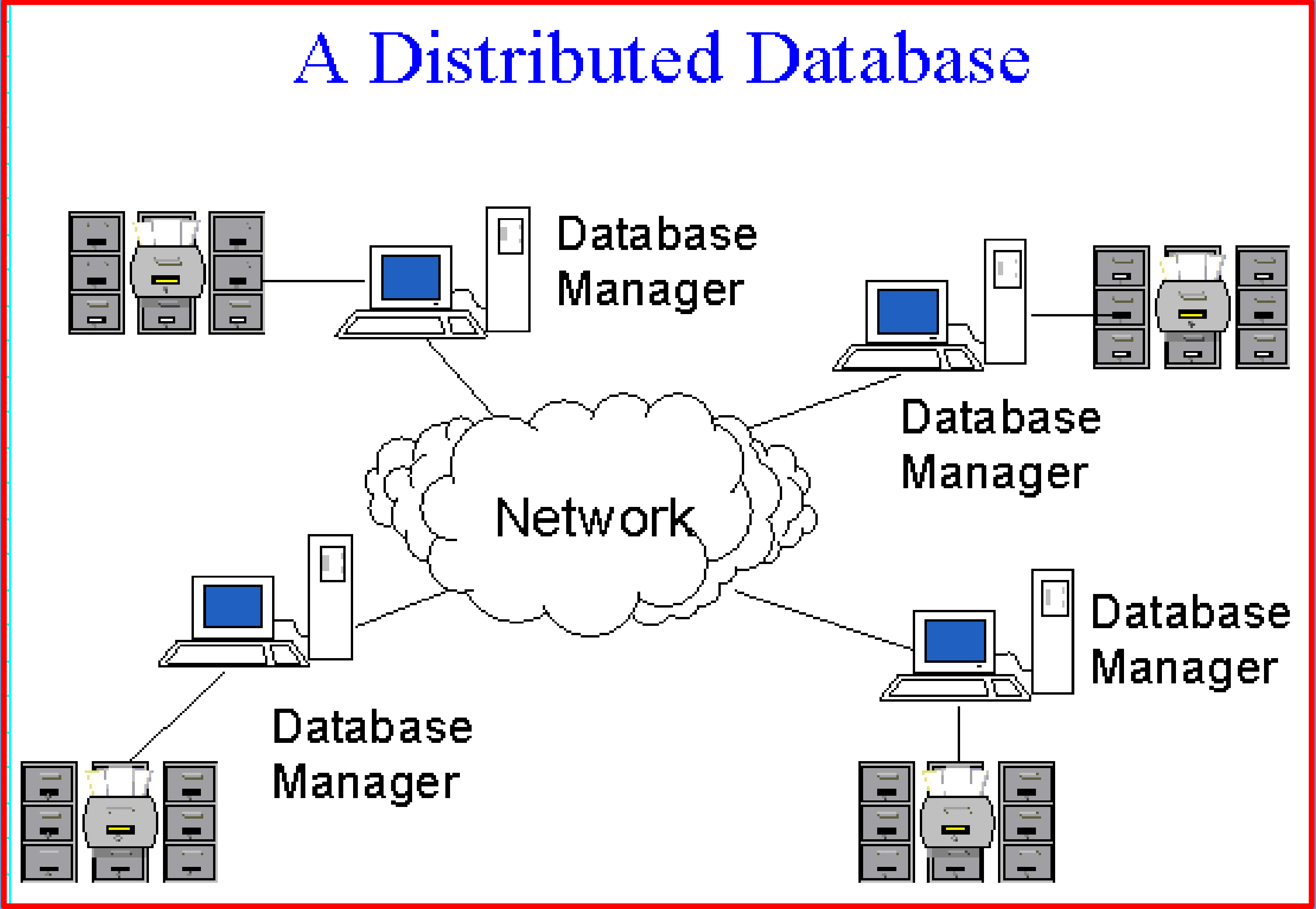
CENTRALIZED DATABASES

This type of data base supports client/server network architecture. Eg- oracle.

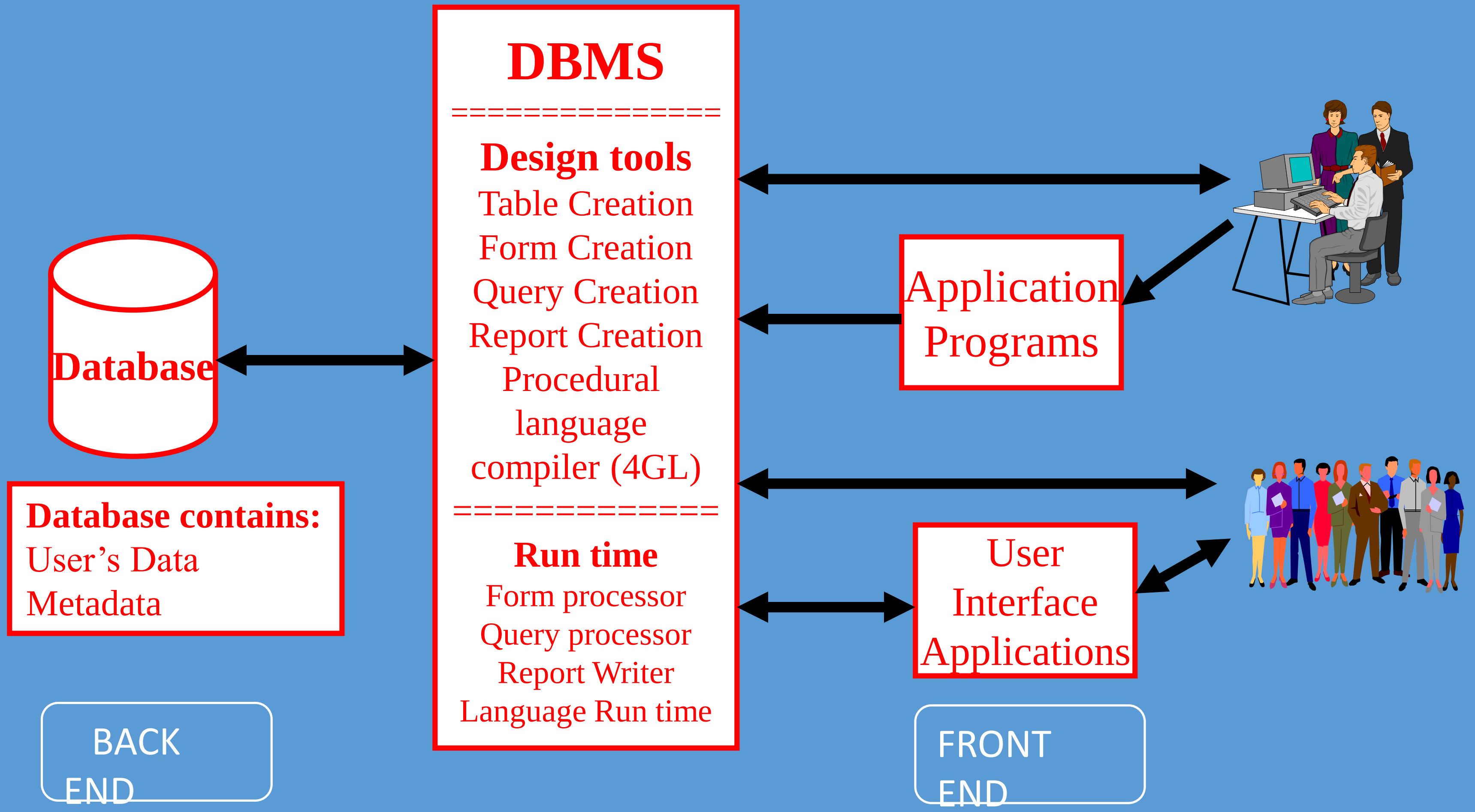


DISTRIBUTED DATABASES

This architecture contains many local servers spread across geographical areas.



DATABASE WORKING MECHANISM



DATABASE COMPONENTS

- Data dictionary A **DBMS** component that stores metadata, contains a list of all files in a database, the number of records in each file.

- Metadata is data about data. It is descriptive information about a particular data set or object.

DATABASE COMPONENTS

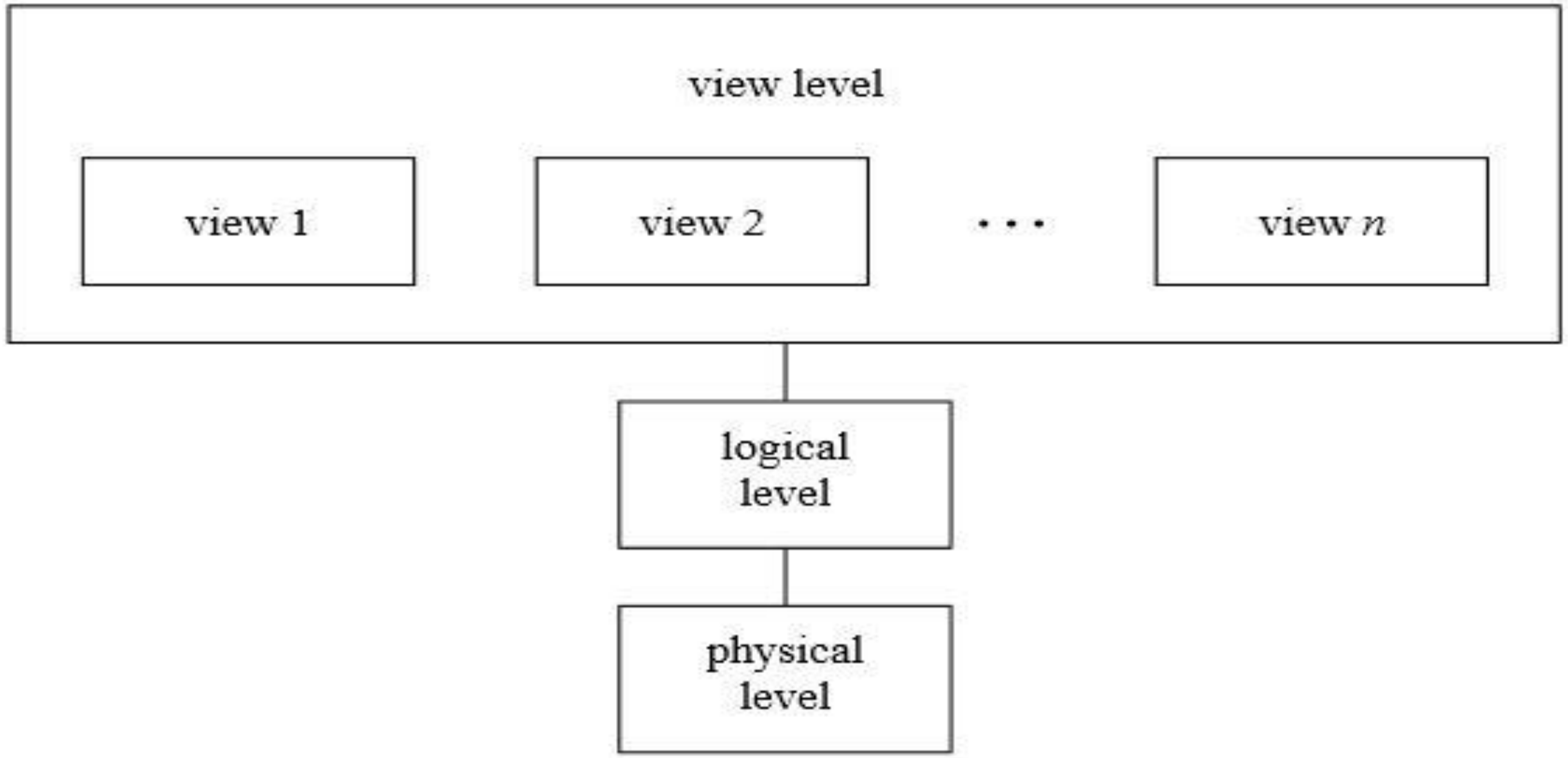
- Query processing refers the retrieval of information from a database.

- Report writer Also called a *report generator,* a program, usually part of a database management system, that extracts information from one or more files and presents the information in a specified format. You can also format data into pie charts, bar charts, and other diagrams.

LEVELS OF DBMS

View of Data

An architecture for a database system



LEVELS OF DBMS

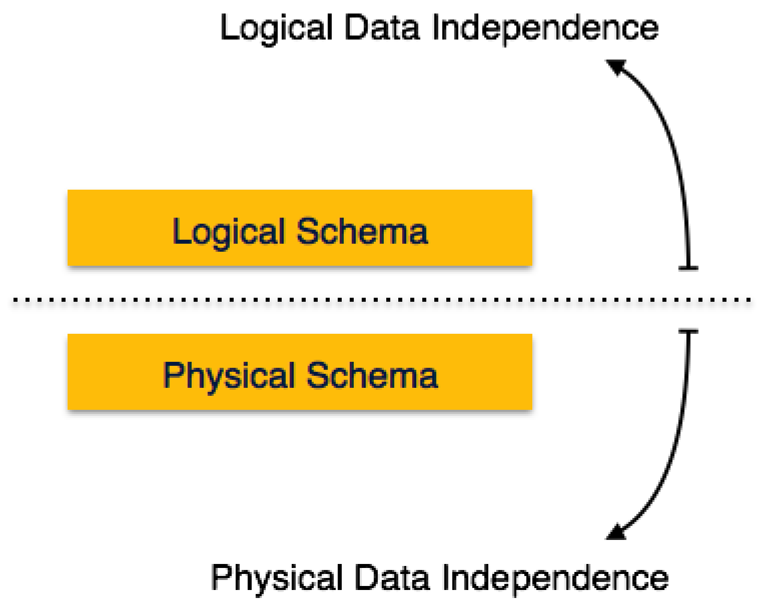
- **Physical level:**
  - describes how a record (e.g., customer) is stored.

- **Logical level:**
  - describes data stored in database, and the relationships among the data.

- **View level:**
  - application programs hide details
    of data types.

DATA INDEPENDENCE

It refers that changing of data in one level does not affect in the data of another next level.



If we do some changes on table format, it should not change the data residing on the disk.

It is the power to change the physical data without impacting the schema or logical data.

INSTANCES AND SCHEMA

- **Schema** – the logical structure of the database
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level

- **Instance** – the actual content of the database at a particular point in time

DATA MODELS

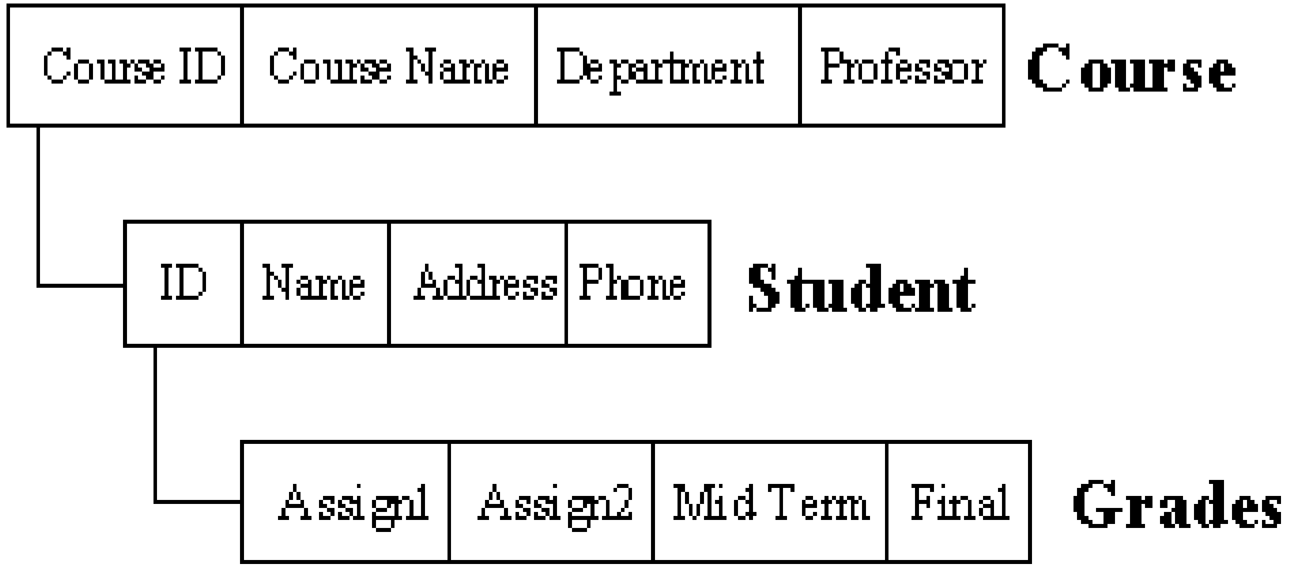
- A collection of tools for describing data and data relationships.

- Types of models :
  - Object-Oriented Model
  - Hierarchical Model
  - Network Model or Codasyl Model
  - E R Model
  - Relational Model

HIERARCHICAL MODEL

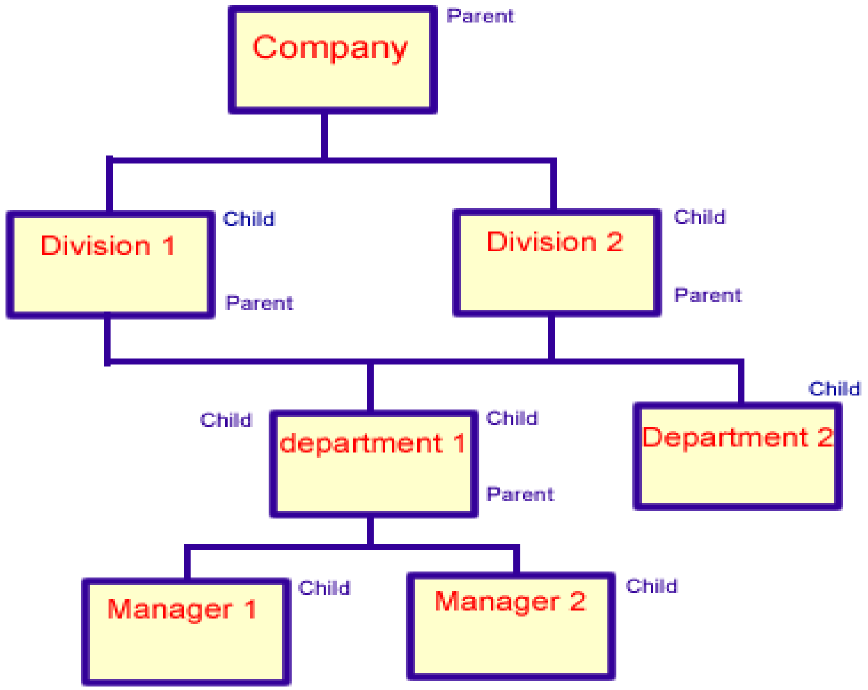
- A hierarchical database is a design that uses a **one-to-many relationship** for data elements. Hierarchical database model uses a tree structure.

Limitation:
Many to Many Relationship
Does not exist.



NETWORK MODEL

Similar to the hierarchical database with the implementation of **many-to-many** relationships.



OBJECT-ORIENTED MODEL

- In the object oriented data model the (OODM), both data and their relationship are contained in a single structure known us an object.

OBJECT-ORIENTED MODEL

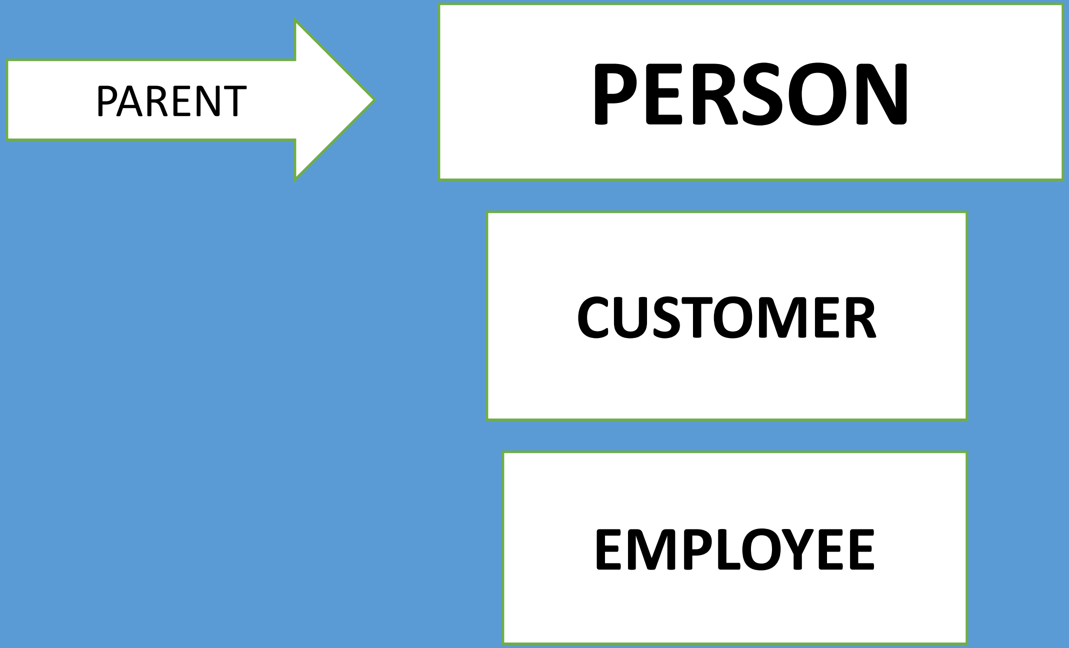
- An object is the abstraction of the real- word entity. An object represents only one occurrence of entity.
- Attributes describe the property of an object.
- Objects that are similar in characteristics are grouped in class.

OBJECT-ORIENTED MODEL

- Class: is a collection of similar objects with shared structure (attributes) and behavior (method)
- Method: represents areal word action such as finding a selected person’s name, changing person’s name or printing a persons address.

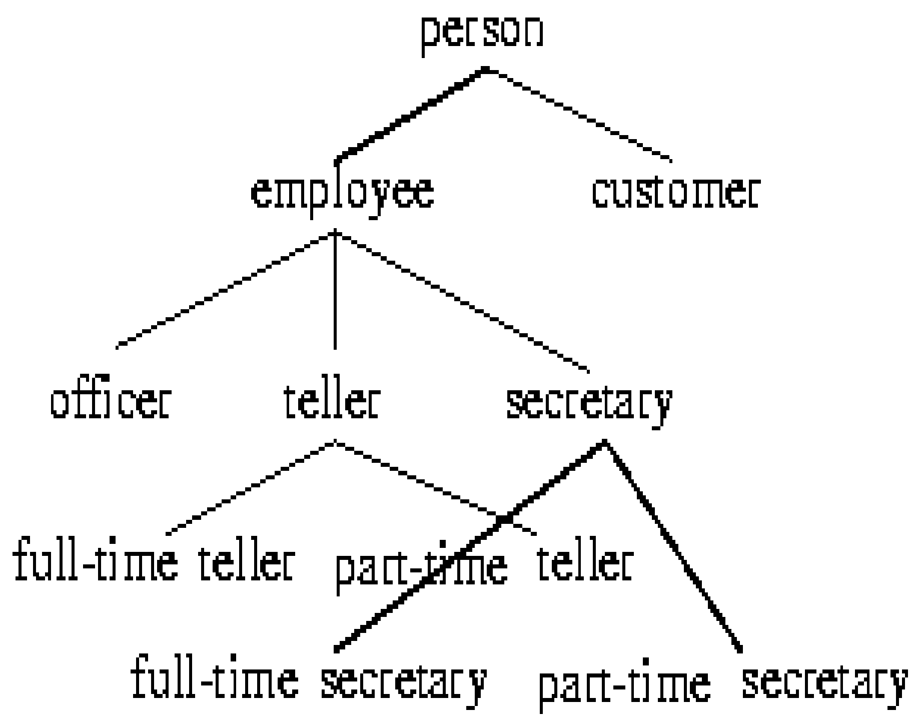
OBJECT-ORIENTED MODEL

- Classes are organized in class hierarchy. The class hierarchy resembles an up side down tree in which each class has only one parent.

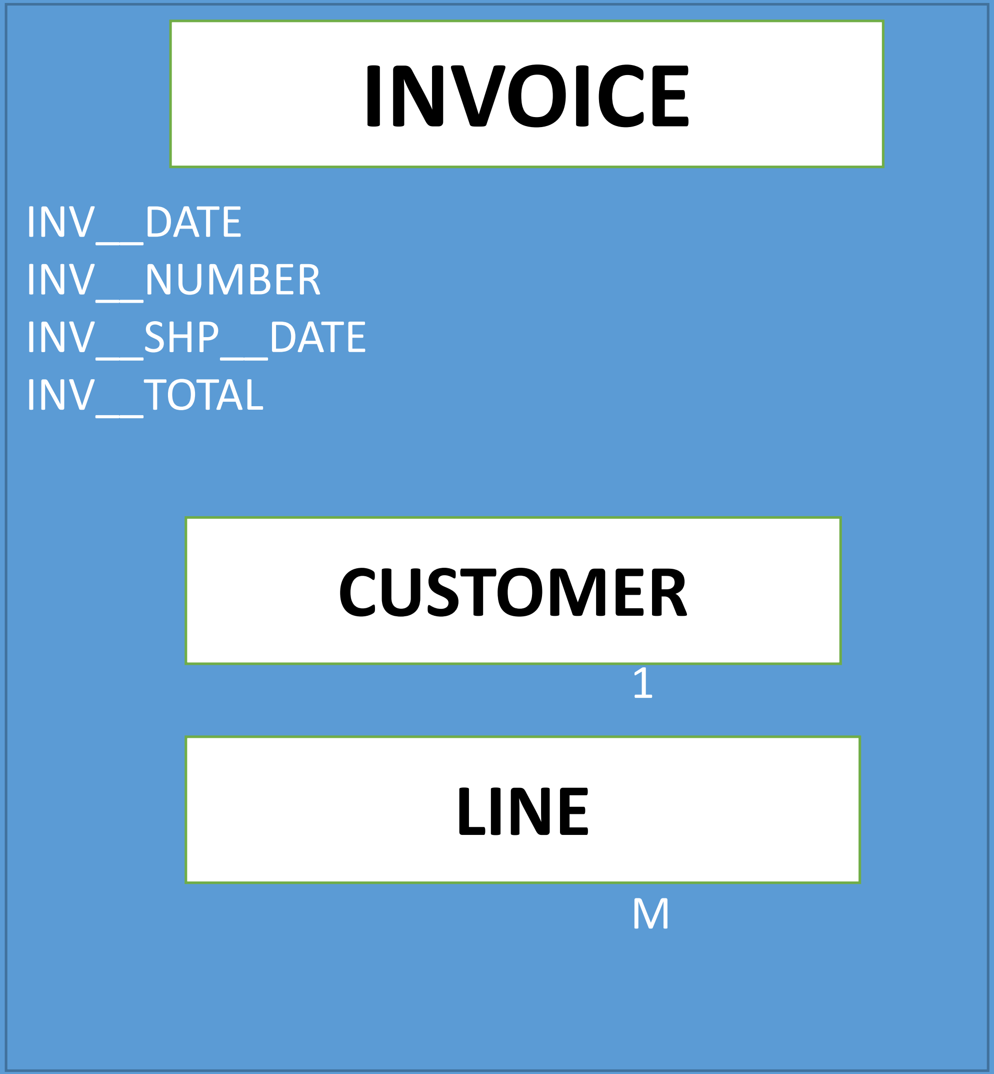


OBJECT-ORIENTED MODEL

- Inheritance is the ability of an object within the class hierarchy to inherit the attributes and methods of the class above it.



OBJECT-ORIENTED MODEL



- Unified modeling language : describes a set of diagrams and symbols that can be used to graphically model a system.

ENTITY RELATIONSHIP MODEL

- The entity-relationship model (or ER model) is a way of graphically representing the logical relationships of entities (or objects) in order to create a database.

- A database can be modeled as:
  - a collection of entities,
  - relationship among entities.

ENTITY RELATIONSHIP MODEL

- ➢ An entity is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- ➢ Entities have attributes
  - Example: a person or a book
- ➢ An entity set is a set of entities of the same type that share the same properties.
  - Example: all persons having an account at a bank.

ENTITY RELATIONSHIP MODEL

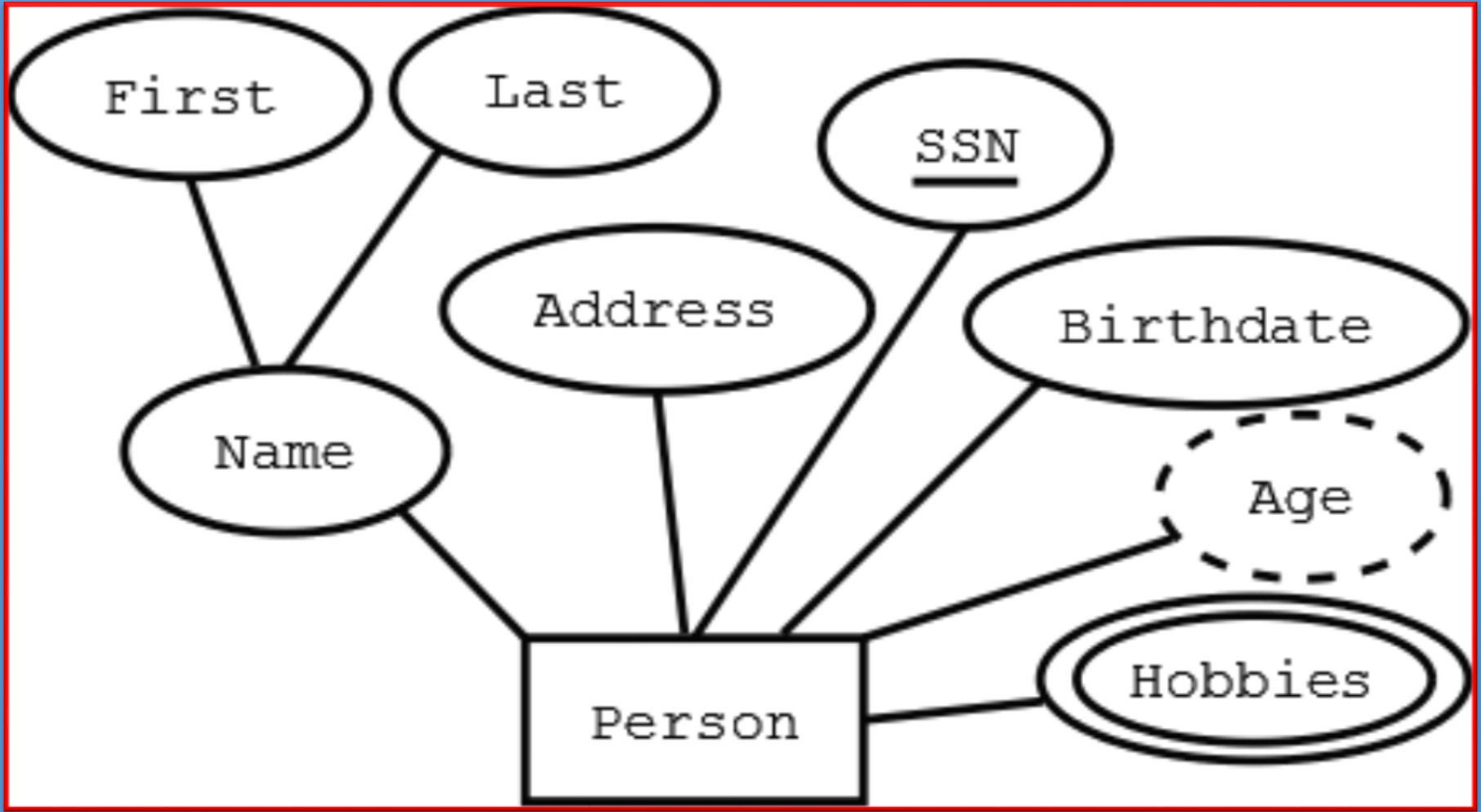
- Domain – the set of permitted values for each attribute

- Attribute types:
  - Simple and composite attributes.
  - Single-valued and multi-valued attributes

    E.g. multivalued attribute: phone-numbers
  - Derived attributes

    Can be computed from other attributes. E.g. age, given date of birth

E-R MODEL : ATTRIBUTE TYPES EXAMPLE



ENTITY RELATIONSHIP MODEL

Degree of Relationship Set

Refers to number of entity set that participate in a relationship set.

- Binary : Relationship set that involve two entity sets.
- Ternary : Relationship sets may involve more than two entity sets.

ENTITY RELATIONSHIP MODEL

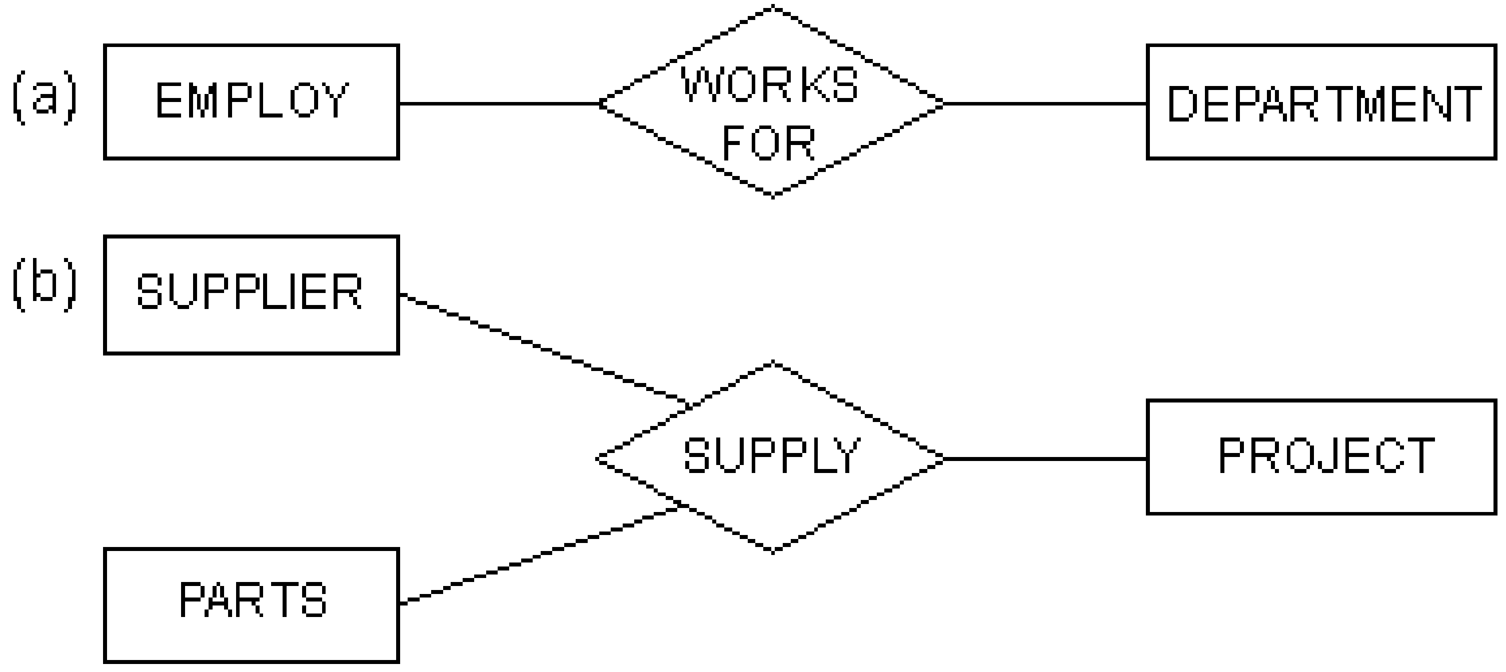


Figure 3.5 : (a) binary relationship WORKS_FOR

(b) ternary relationship SUPPLY

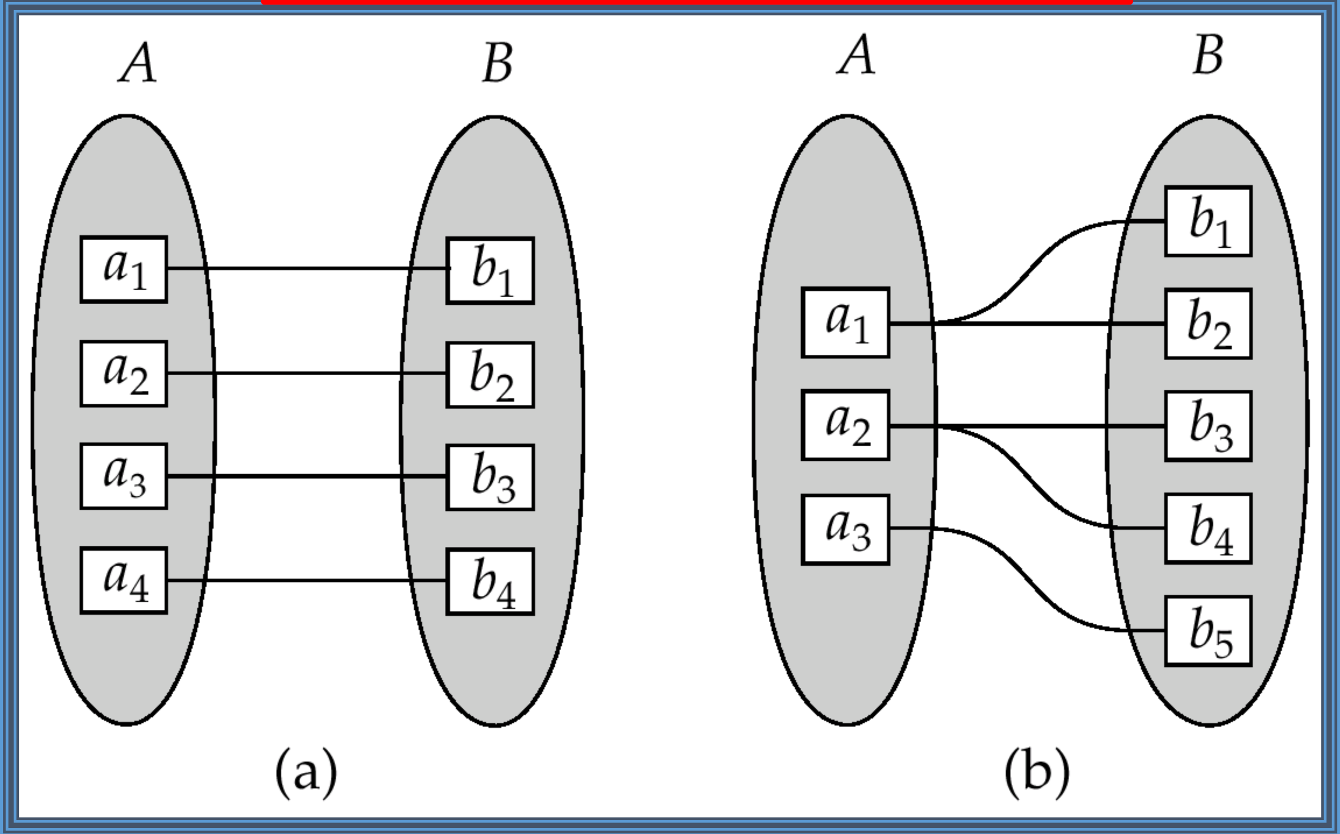
ENTITY RELATIONSHIP MODEL

Mapping Cardinalities

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set, the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

ENTITY RELATIONSHIP MODEL

Mapping Cardinalities



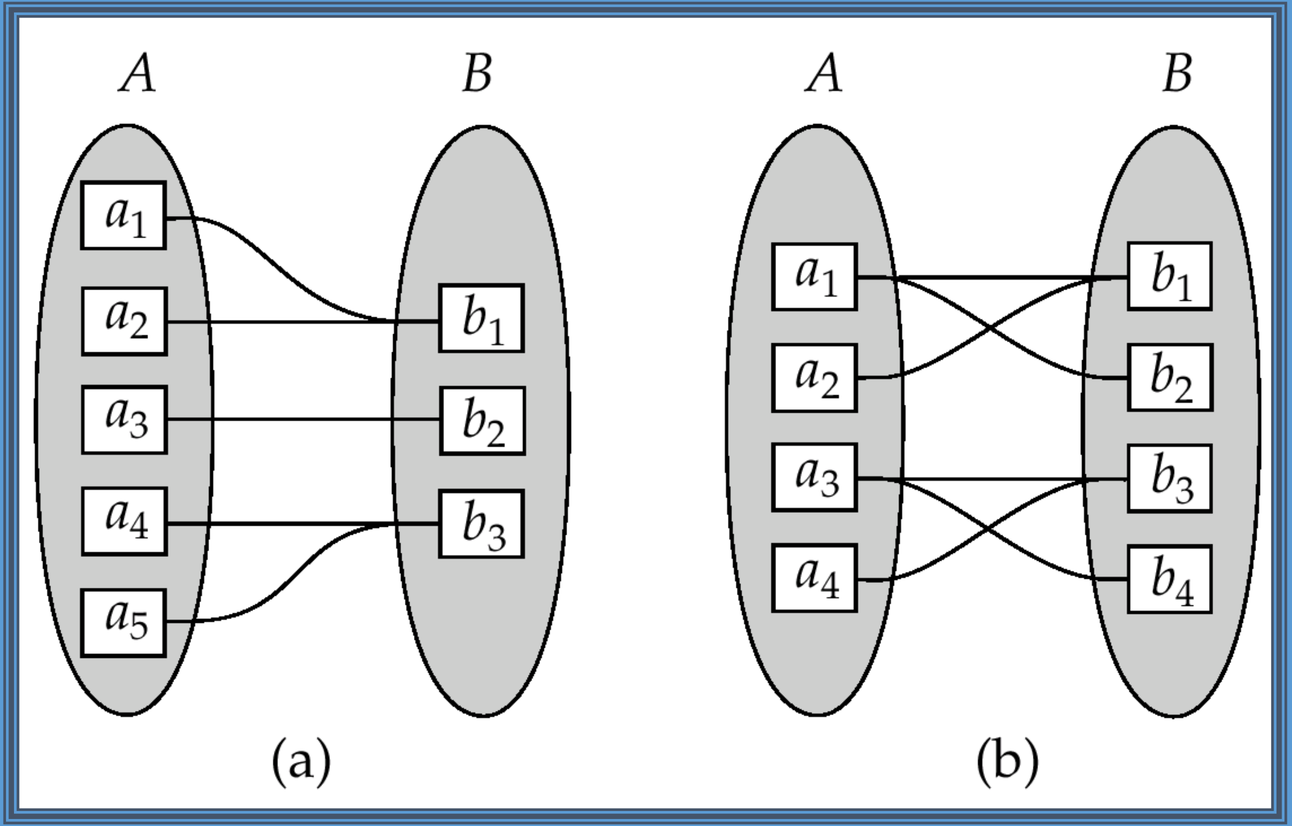
One to one

One to many

Some elements in A and B may not be mapped to any elements in the other set.

ENTITY RELATIONSHIP MODEL

Mapping Cardinalities



Many to one

Many to many

Some elements in A and B may not be mapped to any elements in the other set

E-R MODEL : E-R DIAGRAMS

Rectangles represent entity sets.

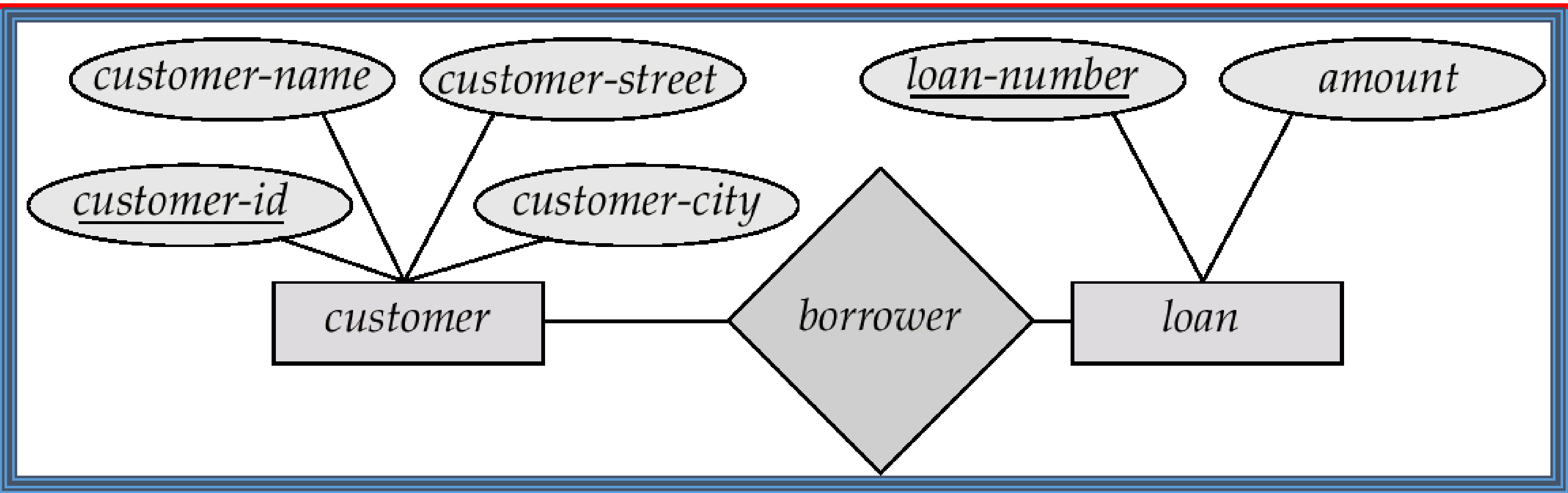
Diamonds represent relationship sets.

Lines link attributes to entity sets and entity sets to relationship sets.

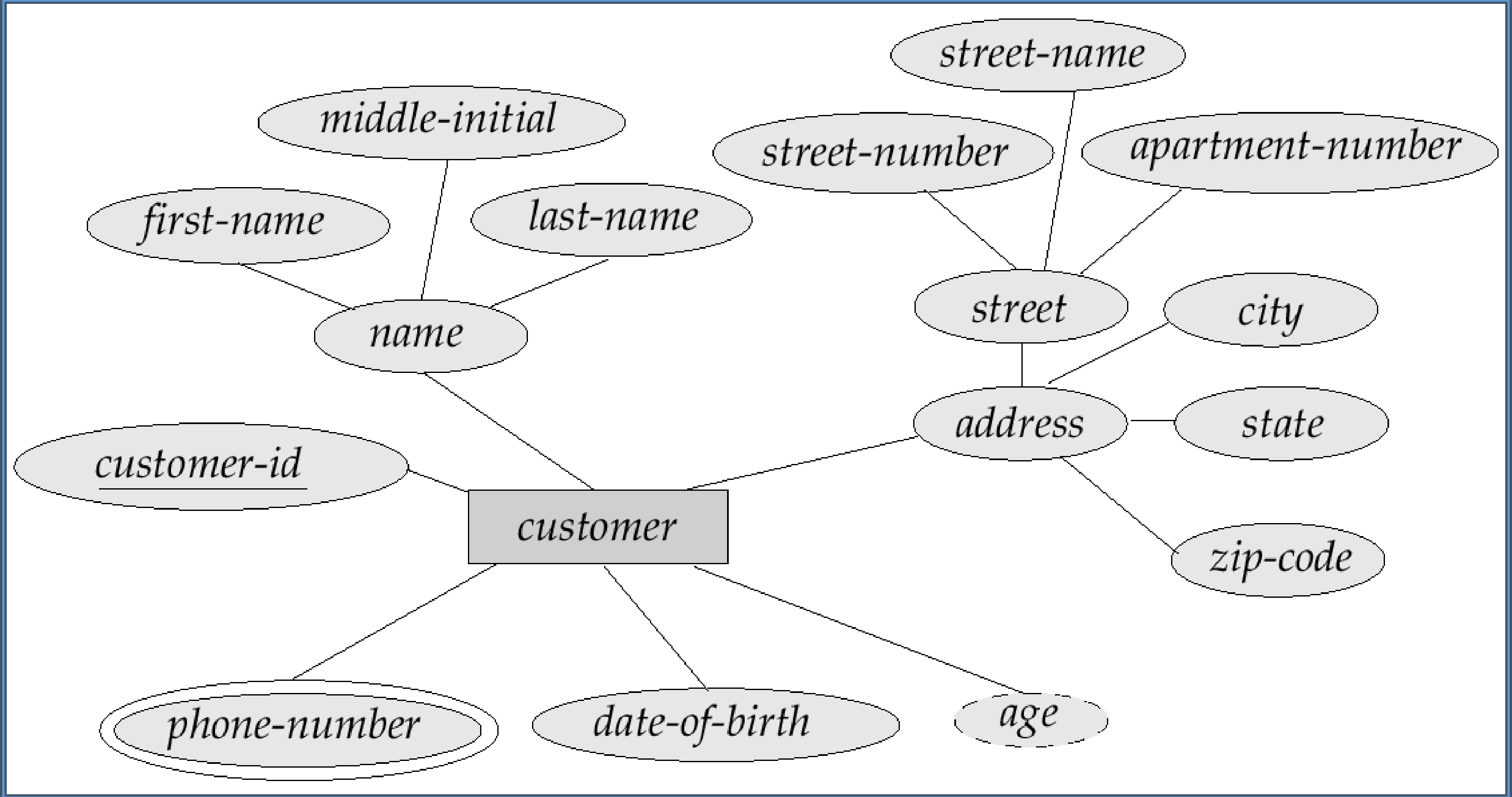
Ellipses represent attributes

- Double ellipses represent multivalued attributes.
- Dashed ellipses denote derived attributes.

Underline indicates primary key attributes



E-R MODEL : E-R DIAGRAMS

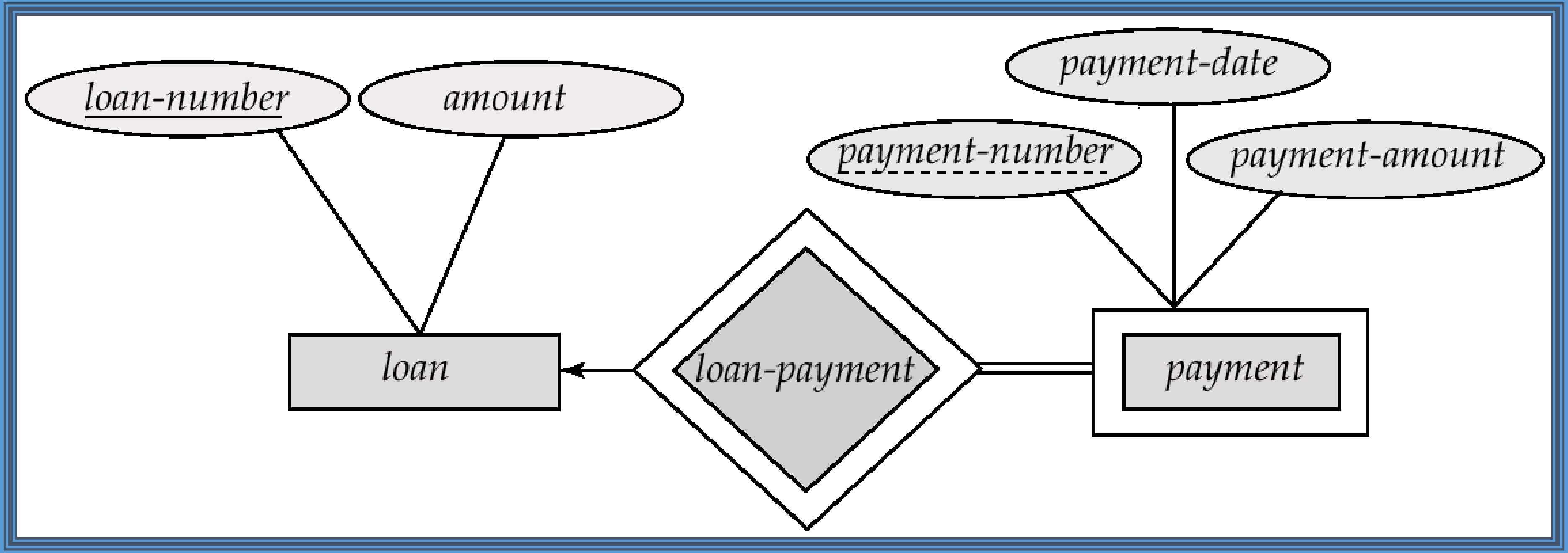


ENTITY RELATIONSHIP MODEL

Weak Entity Set

- An entity set that does not have a primary key is referred to as a weak entity set.
- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dash
- Identifying relationship depicted using a double diamonded line.

E-R MODEL : WEAK ENTITY SET

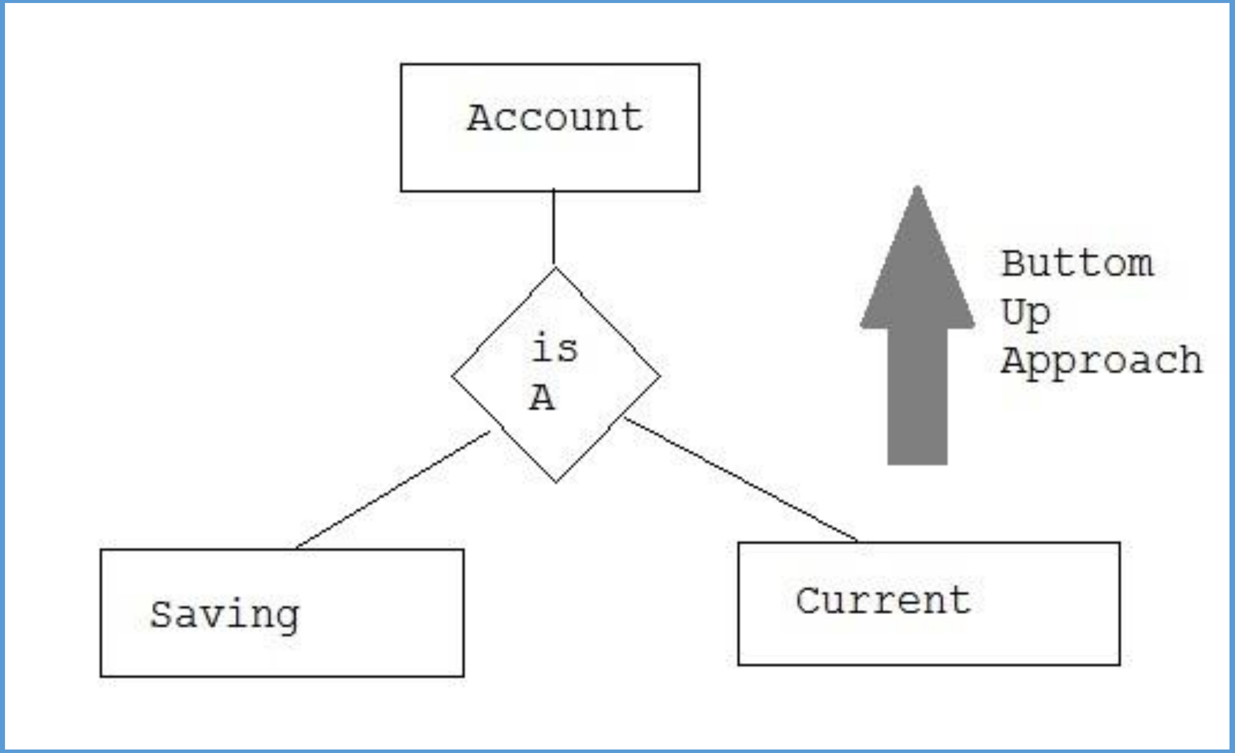


payment-number – discriminator of the *payment* entity set
Primary key for *payment* – (*loan-number, payment-number*)

E-R MODEL : DESIGN SCHEMA

Generalization

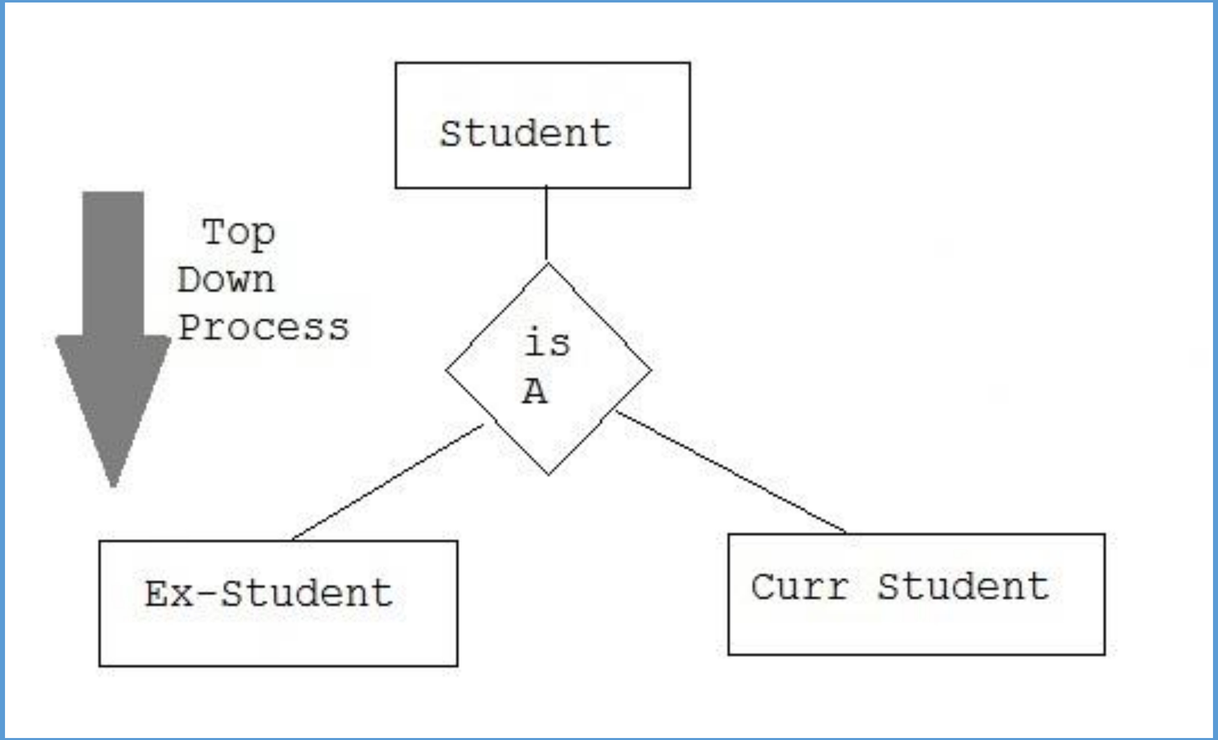
➢It is a bottom-up approach in which two lower level entities combine to form a higher level entity.



E-R MODEL : DESIGN SCHEMA

Specialization

It is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity.

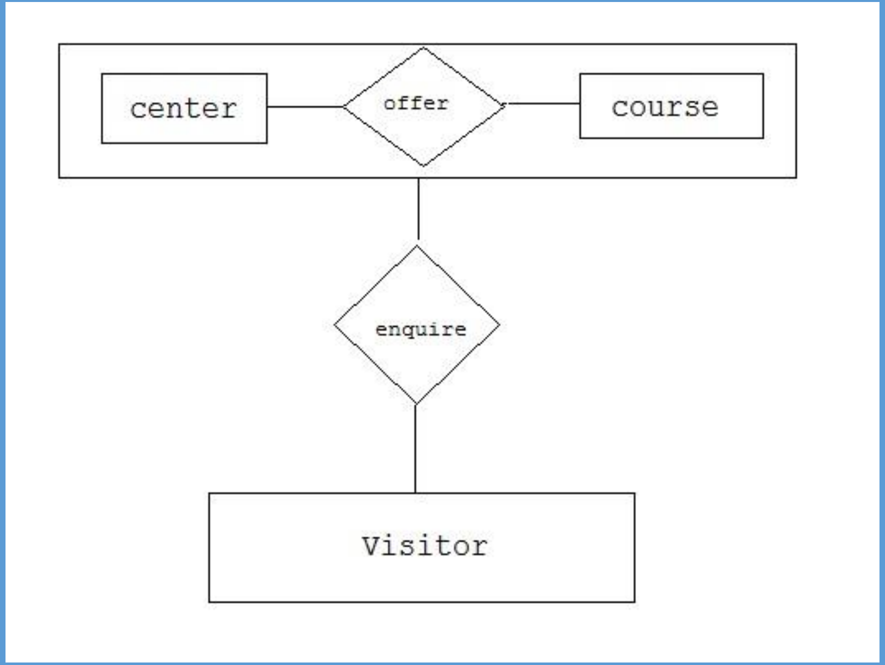


E-R MODEL : DESIGN SCHEMA

Aggregation

It is a process when relation between two entity is treated as a single entity.

Here the relation between Centre and Course, is acting as an Entity in relation with Visitor.



E-R MODEL : KEYS

- Super key is a set of one or more attributes whose values uniquely determine each entity in the entity set.
- Candidate key is a minimal super key in the entity set.

RELATIONAL MODEL

Data stored in tables that are associated by shared attributes (keys).

- Developed by E.F. Codd, C.J. Date (70s)
- Table = Entity = Relation
- Table row = tuple = instance
- Table column = attribute

RELATIONAL MODEL

Attributes

Cust_Id	Cust_Name	Street	City	Acc_NO
1000	Ravi	Lohia marg	gorakhpur	A-05
1001	Shobhit	Ashok marg	Lucknow	A-10
1002	Gaurav	Gokhle marg	Faridabad	A-11
1003	Arindom	Pk sen road	Kanpur	A-12
1004	Neeraj	Cantt road	chandigarh	A-13

tuples

RELATIONAL DATABASE DEFINITIONS

- Entity
  - Object, Concept or event (subject)
- Attribute
  - a Characteristic of an entity
- Tuple
  - represents a record
- Degree
  - total number of attributes
- Cardinality
  - total number of tuples

DATABASE TABLE KEYS

A key of a relation is a subset of attributes with the following attributes:

- Unique identification
- Non-redundancy

DATABASE TABLE KEYS

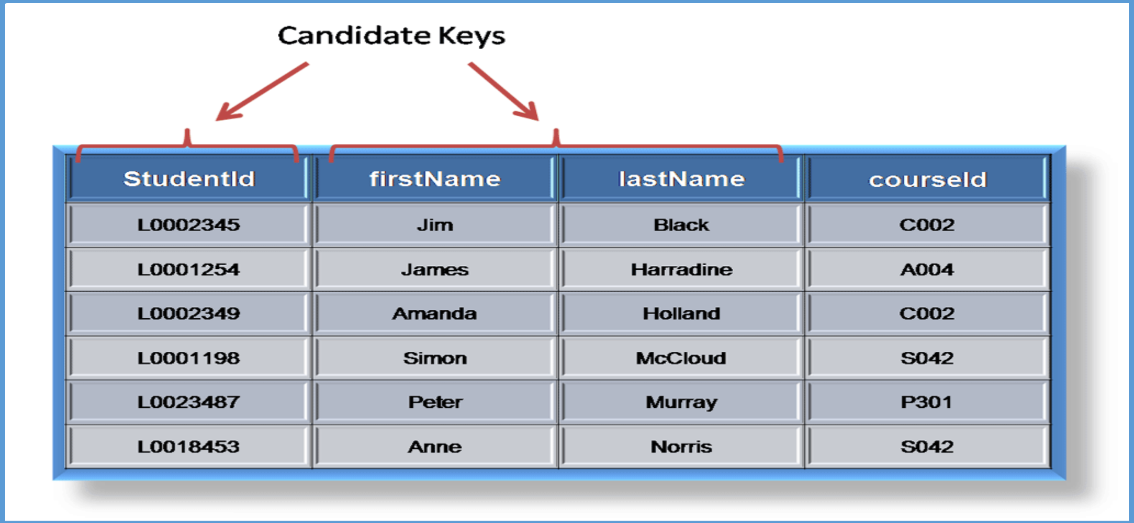
Super Key

Super Key is defined as a set of attributes within a table that uniquely identifies each record within a table. Super Key is a superset of Candidate key.

DATABASE TABLE KEYS

Candidate Key

Candidate keys are defined as the set of fields from which primary key can be selected. It is an attribute or set of attribute that can act as a primary key for a table to uniquely identify each record in that table.


Candidate Keys

StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

TYPES OF KEYS

Primary key

It is used to identify a record uniquely in a database

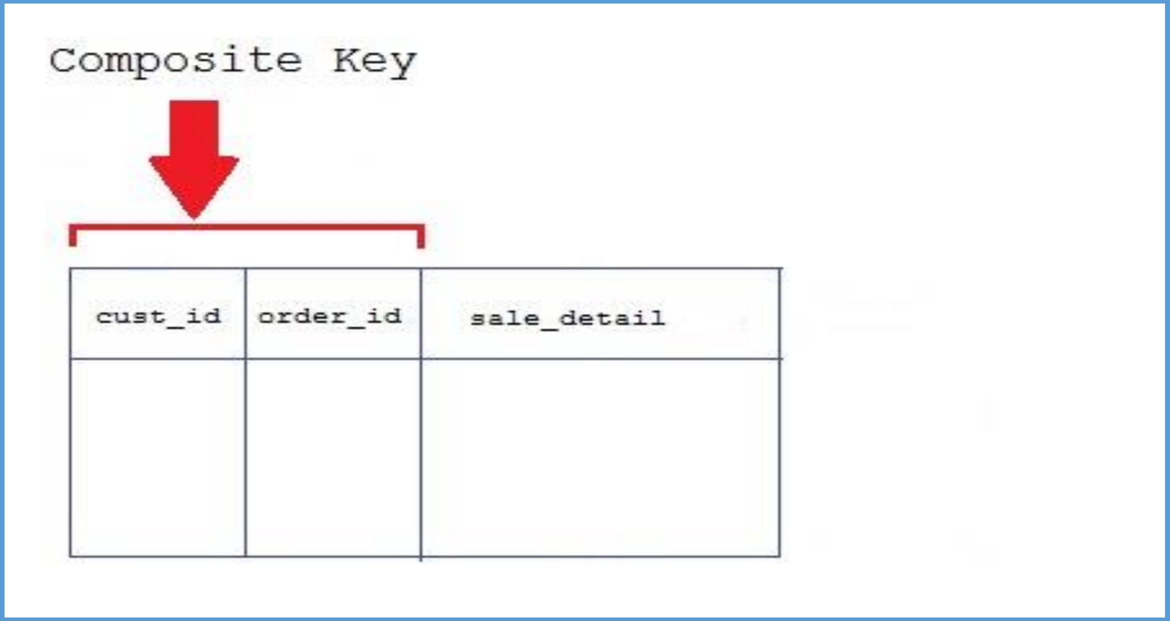
Secondary or Alternative key

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

TYPES OF KEYS

Composite Key

Key that consist of two or more attributes that uniquely identify an entity occurrence is called **Composite key**.



TYPES OF KEYS

Foreign Key

A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second.

Table A

Name	Address	Parcel #
John Smith	18 Lawyers Dr.	756554
T. Brown	14 Summers Tr.	887419

Table B

Parcel #	Assessed Value
887419	152,000
446397	100,000

REFERENTIAL INTEGRITY

- **Referential integrity** concerns two or more tables that are related.

- Example: IF **table A** contains a foreign key that matches the primary key of **table B** THEN values of this foreign key either match the value of the primary key for a row in **table B** or must be *null*.

Thank You
shift
4

WHAT IS DATABASE NORMALIZATION ?

- Database Normalization is a technique of organizing the data in the database.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization is used for mainly two purpose,

  Eliminating redundant (useless) data.

  Ensuring data dependencies make sense i.e data is logically stored.

PROBLEM WITHOUT NORMALIZATION!

- Without Normalization, it becomes difficult to handle and update the database, without facing data loss.

- Insertion, Updation and Deletion Anomalies are very frequent if Database is not Normalized.

To understand these anomalies

- let us take an example of **Student** table in next Slide.

LET US TAKE EXAMPLE:

S_id	S_Name	S_Address	Subject_Opted
S01	Mr. A	Gwalior	Information Technology
S02	Mr. B	Bhopal	Cyber Security
S03	Mr. C	Lucknow	Computer Science
S04	Mr. D	Ambala	Mechanics
S05	Mr. C	Lucknow	Microprocessors

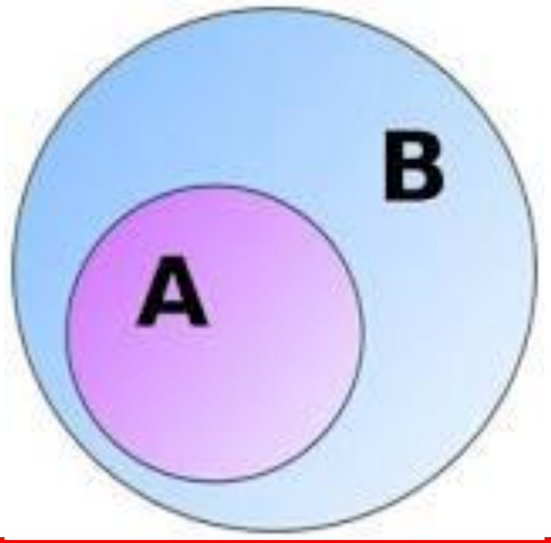
EXPLANATION

- **Updation anomaly :** To update address of a student who occurs twice or more than twice in a table, we will have to update **S_Address** column in all the rows, else data will become inconsistent.
- **Insertion anomaly :** Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert **NULL** there, leading to Insertion anomaly.
- **Deletion anomaly :** If (S_id) S01 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

BEFORE STARTING DATABASE NORMALIZATION RULE, IT IS IMPORTANT TO UNDERSTAND THESE SOME BASIC TERMINOLOGIES

Meaning of subset and superset

A is a proper **subset** of B and conversely B is a proper **superset** of A. In mathematics, especially in set theory, if A is "contained" inside B, that is, all elements of A are also elements of B. A and B may coincide.



Database Keys

Keys are very important part of Relational database. They are used to establish and identify relation between tables. They also ensure that each record within a table can be uniquely identified by combination of one or more fields within a table.

- **Super Key**

Super Key is defined as a set of attributes within a table that uniquely identifies each record within a table. Super Key is a superset of Candidate key.

- **Candidate Key**

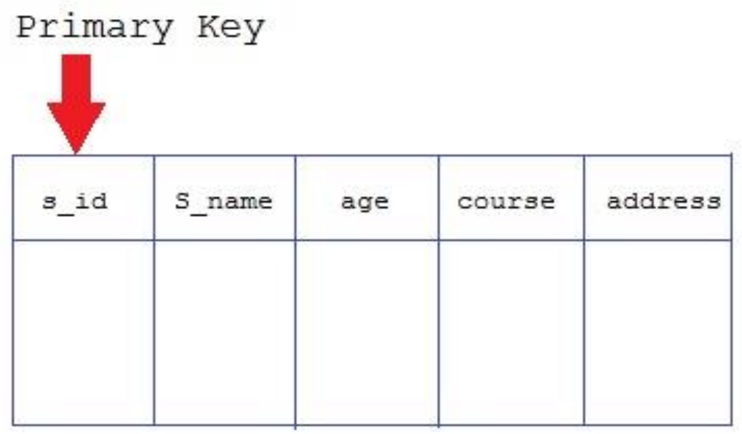
Candidate keys are defined as the set of fields from which primary key can be selected. It is an attribute or set of attribute that can act as a primary key for a table to uniquely identify each record in that table.

Here **Non-prime** Attributes are attributes other than **Primary attribute**.

CONTINUE..

- **Primary Key**

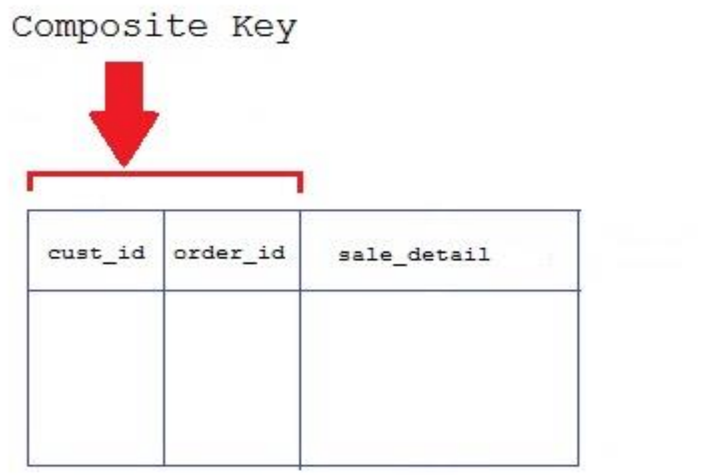
Primary key is a candidate key that is most appropriate to become main key of the table. It is a key that uniquely identify each record in a table.



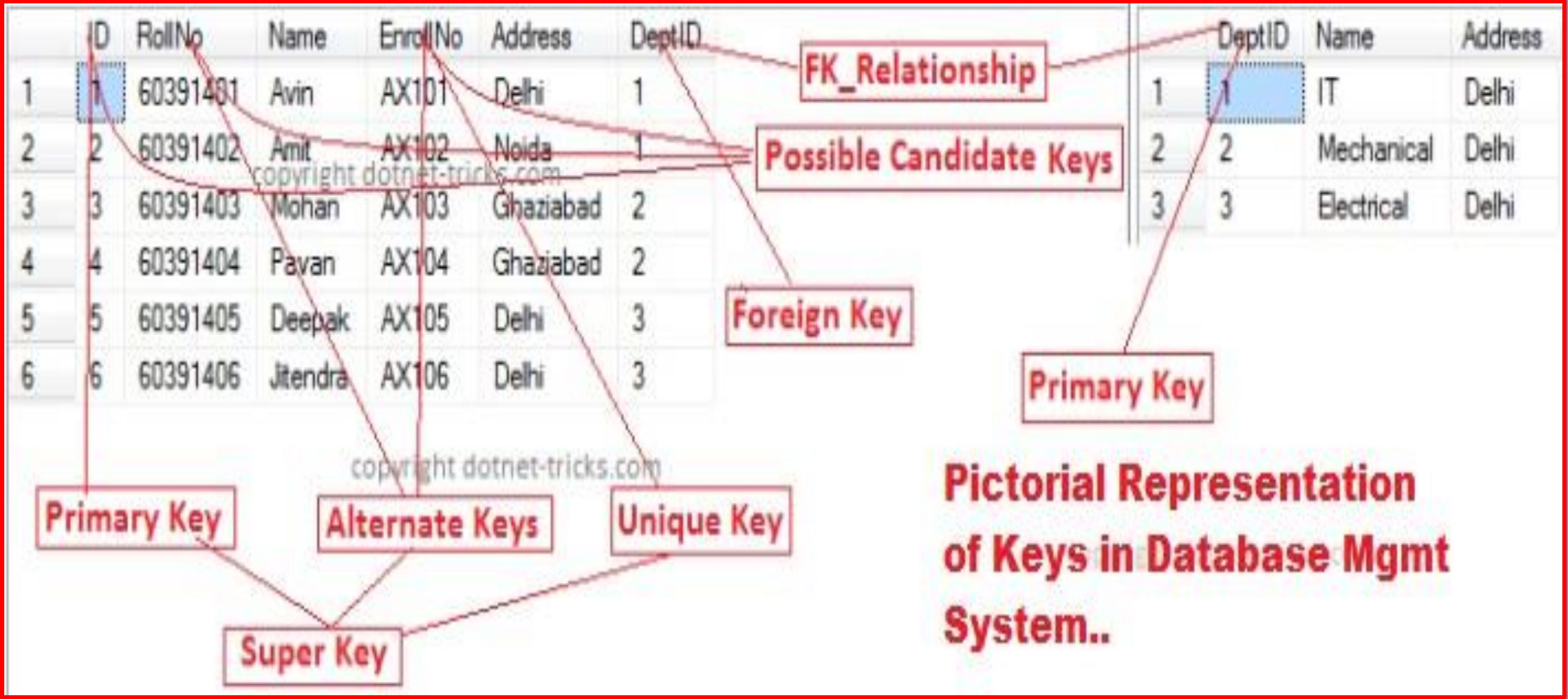
CONTINUE..

- **Composite Key**

Key that consist of two or more attributes that uniquely identify an entity occurance is called **Composite key**. But any attribute that makes up the **Composite key** is not a simple key in its own.



OVERALL KEY SUMMARY..



	ID	RollNo	Name	EnrollNo	Address	DeptID
1	1	60391401	Avin	AX101	Delhi	1
2	2	60391402	Amit	AX102	Noida	1
3	3	60391403	Mohan	AX103	Ghaziabad	2
4	4	60391404	Pavan	AX104	Ghaziabad	2
5	5	60391405	Deepak	AX105	Delhi	3
6	6	60391406	Jitendra	AX106	Delhi	3

	DeptID	Name	Address
1	1	IT	Delhi
2	2	Mechanical	Delhi
3	3	Electrical	Delhi

Pictorial Representation of Keys in Database Mgmt System..

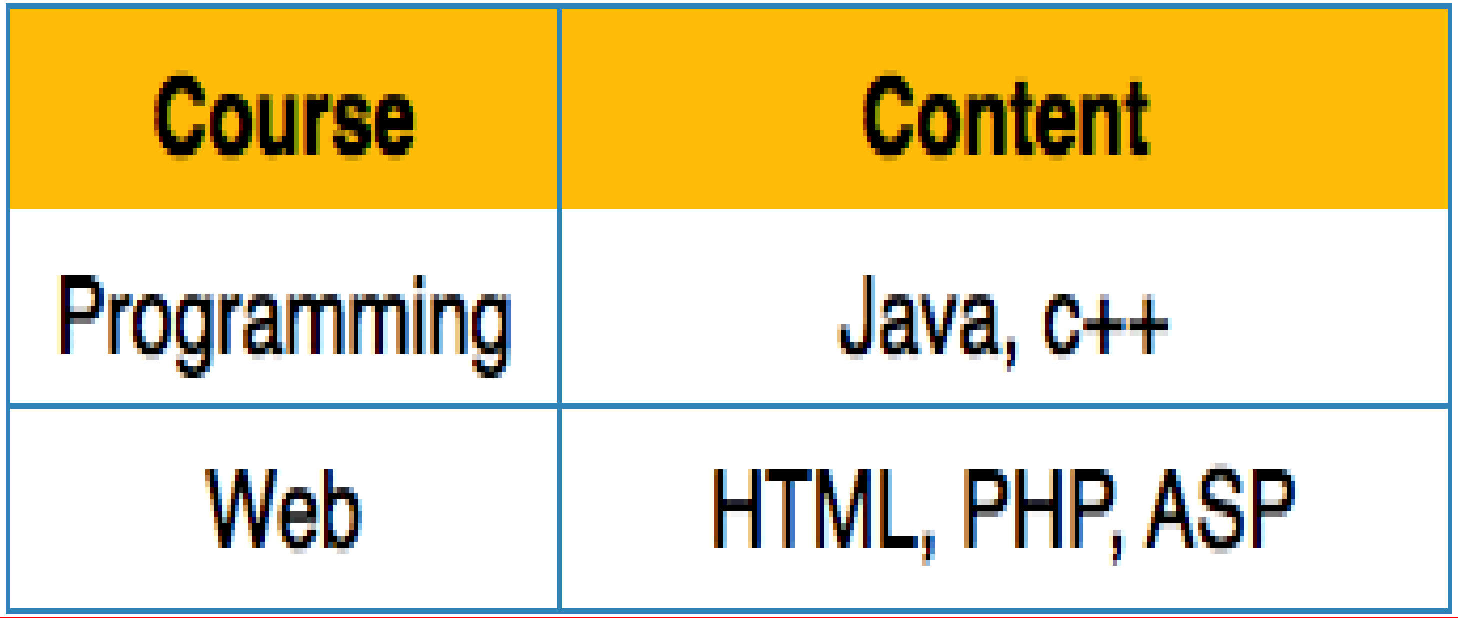
NORMALIZATION RULE

Normalization rule are divided into following normal form.

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF

FIRST NORMAL FORM

- First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

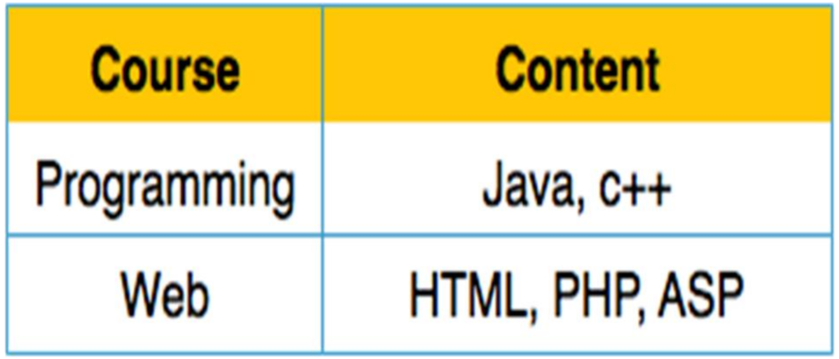


Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

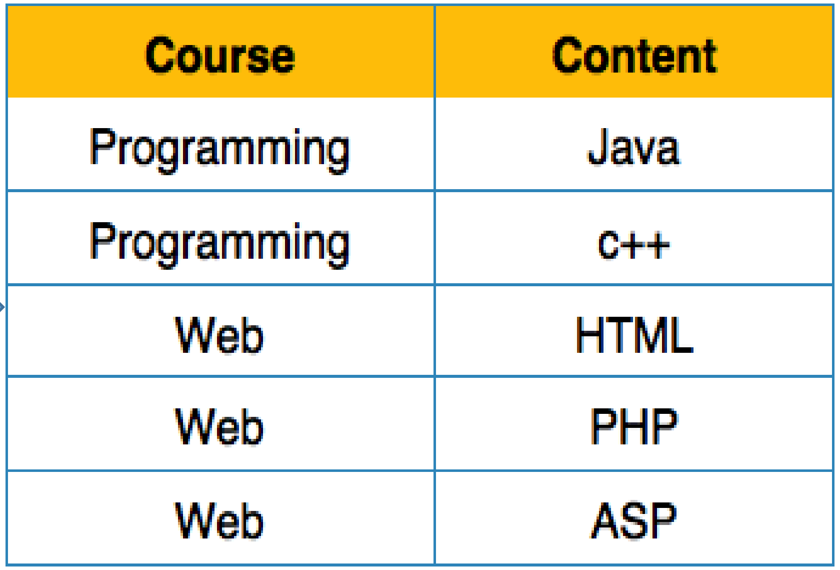
CONTINUE..

- We re-arrange the relation (table) as below, to convert it to First Normal Form.

Each attribute must contain only a single value from its pre-defined domain.



Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP



Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

SECOND NORMAL FORM

- Before we learn about the second normal form, we need to understand the following −

Prime attribute − An attribute, which is a part of the prime-key, is known as a prime attribute.

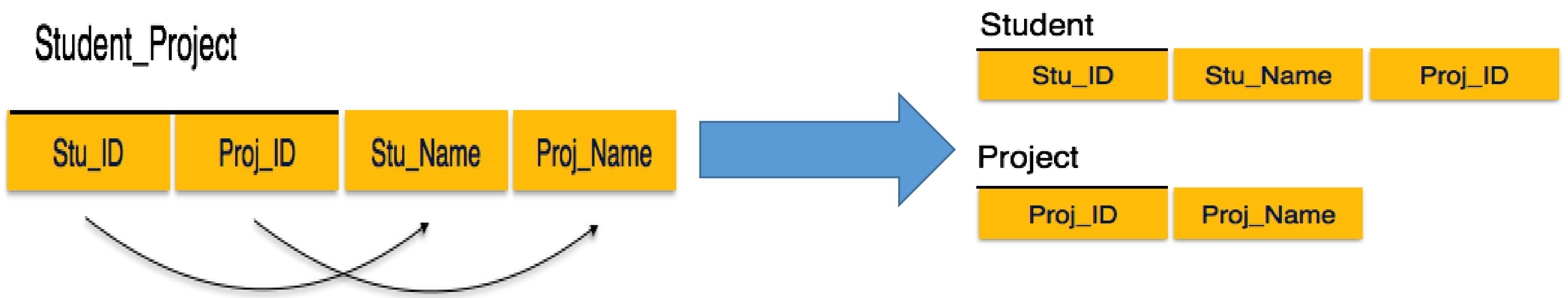
Non-prime attribute − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

- **Functional Dependency -** Functional dependency is a relationship that exists when one attribute uniquely determines another attribute.

If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X->Y, which specifies Y is functionally dependent on X.

CONTINUE..

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute.



EXPLANATION…

- We see here in Student_Project relation that the prime key attributes are Stu_ID and Proj_ID. According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually. But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently. This is called **partial dependency**, which is not allowed in Second Normal Form.

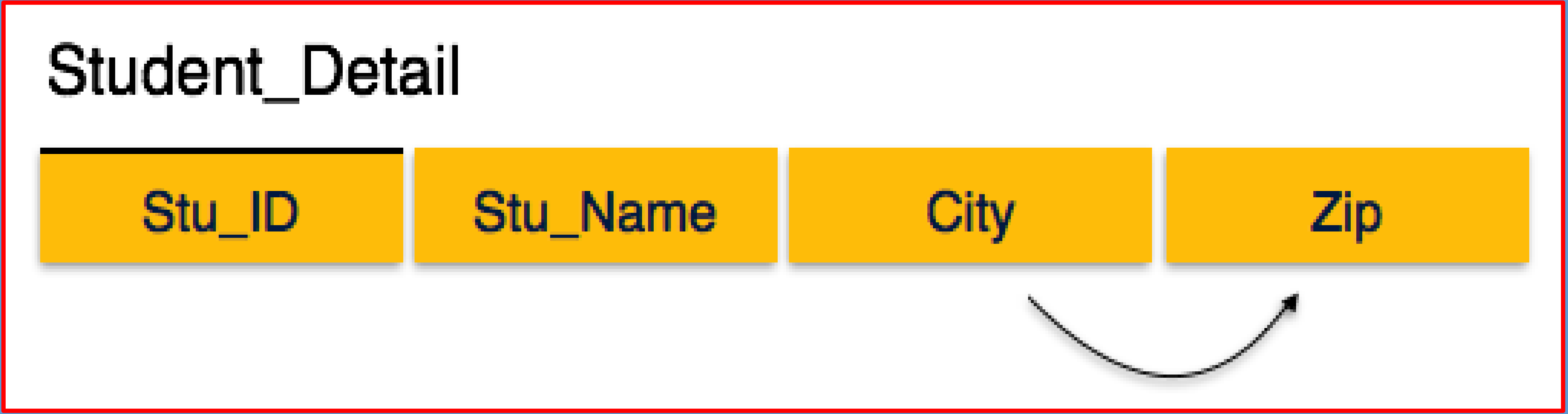
We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

THIRD NORMAL FORM

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → A, then either −
  - X is a superkey or,
  - A is prime attribute.

CONTINUE..



EXPLANATION..

- We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute. We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.

- To bring this relation into third normal form, we break the relation into two relations as follows −

Student_Detail

Stu_ID	Stu_Name	Zip

ZipCodes

Zip	City

ARMSTRONG'S AXIOMS

- If F is a set of functional dependencies then the closure of F, denoted as F^+, is the set of all functional dependencies logically implied by F. Armstrong's Axioms are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.
- **Reflexive rule** − If alpha is a set of attributes and beta is_subset_of alpha, then alpha holds beta.
- **Augmentation rule** − If a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.
- **Transitivity rule** − Same as transitive rule in algebra, if a → b holds and b → c holds, then a → c also holds. a → b is called as a functionally that determines b.

TRIVIAL FUNCTIONAL DEPENDENCY

- Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.
- **Trivial** − If a functional dependency (FD) X → Y holds, where Y is a subset of X, then it is called a trivial FD.
- **Non-trivial** − If an FD X → Y holds, where Y is not a subset of X, then it is called a non-trivial FD.
- **Completely non-trivial** − If an FD X → Y holds, where x intersect Y = Φ, it is said to be a completely non-trivial FD.

BCNF

- Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that −
- For any non-trivial functional dependency, X → A, X must be a super-key.
- In the above image, Stu_ID is the super-key in the relation Student_Detail and Zip is the super-key in the relation ZipCodes. So,

Stu_ID → Stu_Name, Zip

and

Zip → City

Which confirms that both the relations are in BCNF.

FORMAL DEFINITIONS OF THE NORMAL FORMS

- 1st Normal Form (1NF)

Def: A table (relation) is in 1NF if

1. There are no duplicated rows in the table.
2. Each cell is single-valued (i.e., there are no repeating groups or arrays).
3. Entries in a column (attribute, field) are of the same kind.

CONTINUE..

- **2nd Normal Form (2NF)**

Def: A table is in 2NF if it is in 1NF and if all non-key attributes are dependent on all of the key.

- Note: Since a partial dependency occurs when a non-key attribute is dependent on only a part of the (composite) key, the definition of 2NF is sometimes phrased as, "A table is in 2NF if it is in 1NF and if it has no partial dependencies."

CONTINUE..

- 3rd Normal Form (3NF)

Def: A table is in 3NF if it is in 2NF and if it has no transitive dependencies.

- Boyce-Codd Normal Form (BCNF)

Def: A table is in BCNF if it is in 3NF and if every determinant is a candidate key.

DBMS - TRANSACTION ACID PROPERTIES

- A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain:

- **A**tomicity, **C**onsistency, **I**solation, and **D**urability − commonly known as ACID properties − in order to ensure accuracy, completeness, data integrity.

ATOMICITY

- This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none.

Explanation:

There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

CONSISTENCY

- The database must remain in a consistent state after any transaction.

Explanation:

No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

ISOLATION

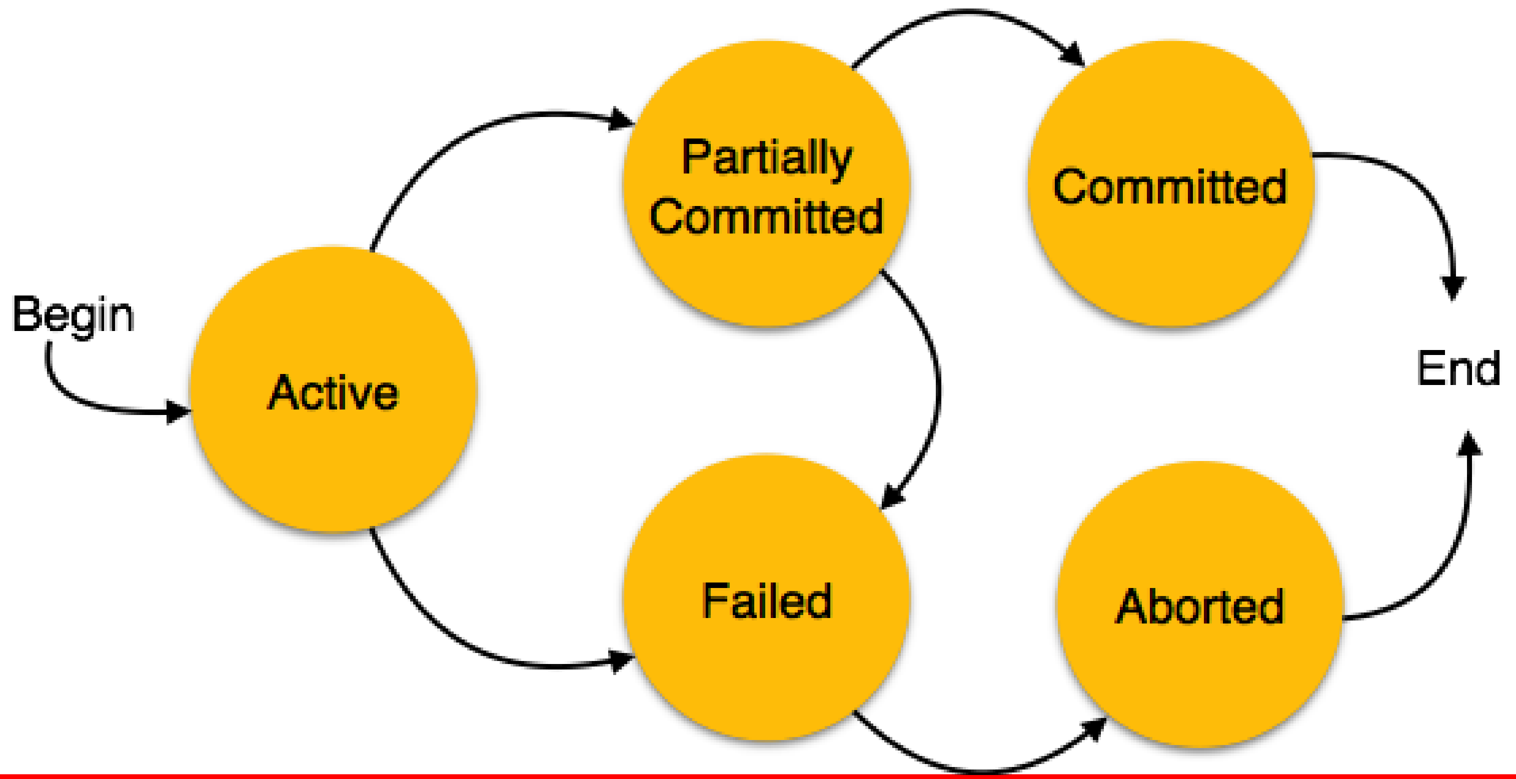
- In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

DURABILITY

- The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

STATES OF TRANSACTIONS

- A transaction in a database can be in one of the following states –



ELABORATION

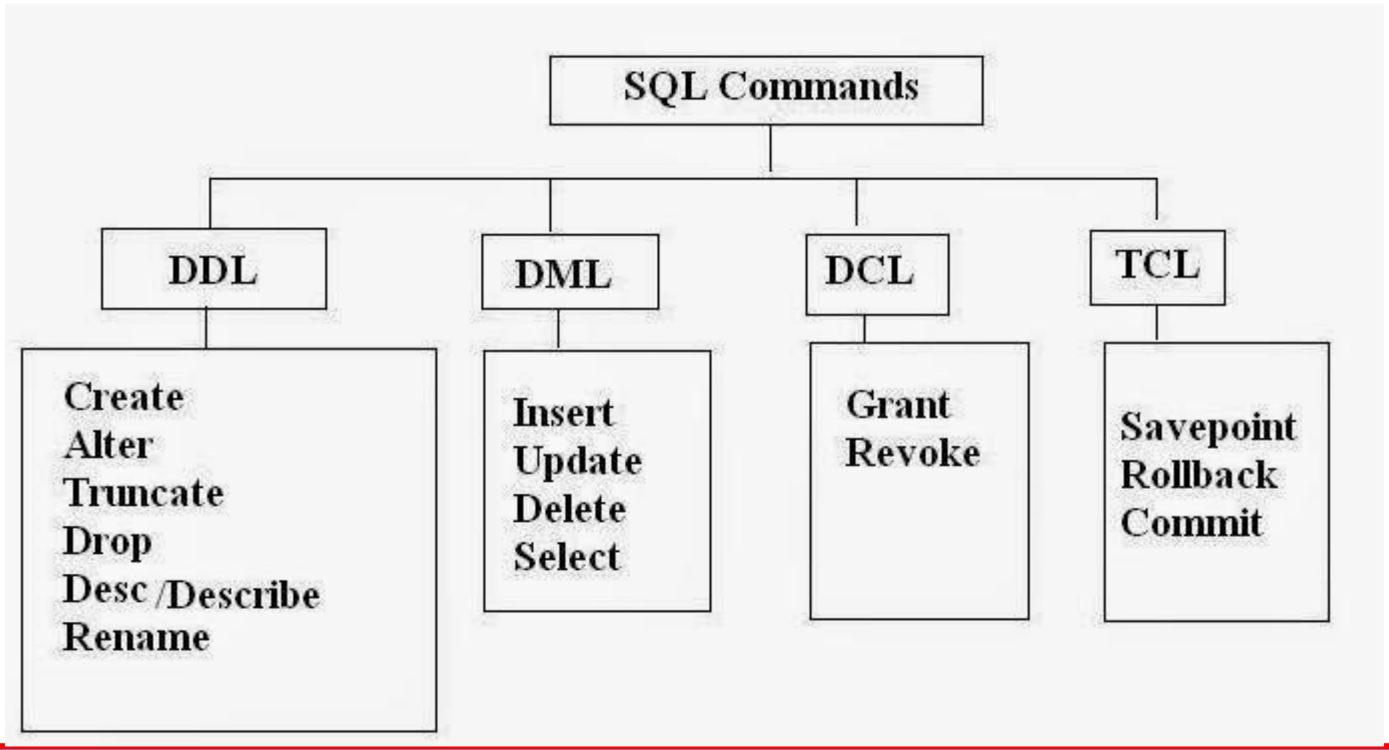
- **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts −
  - **Re-start the transaction**
  - **Kill the transaction**
- **Committed** − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

INTRODUCTION TO SQL

- Structure Query Language(SQL) is a programming language used for storing and managing data in RDBMS.
- SQL was the first commercial language introduced for E.F Codd's **Relational** model.
- Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) uses **SQL** as the standard database language. SQL is used to perform all type of data operations in RDBMS.

SQL COMMANDS

- SQL defines following data languages to manipulate data of RDBMS.



DDL : DATA DEFINITION LANGUAGE

Data Definition Language (DDL) statements are used to define the database structure or schema. Some examples:

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

CREATE COMMAND

- **create command**

create is a DDL command used to create a table or a database.

- **Creating a Database**

To create a database in RDBMS, *create* command is uses. Following is the Syntax,

create database *database-name*;

Example for Creating Database

- create database Test; The above command will create a database named **Test**.

ALTER COMMAND

- alter command is used for alteration of table structures. There are various uses of alter command, such as,

to add a column to existing table

to rename any existing column

to change data type of any column or to modify its size.

alter is also used to drop a column.

TO ADD COLUMN TO EXISTING TABLE

- Using alter command we can add a column to an existing table. Following is the Syntax,

alter table *table-name* add(**column-name** *datatype*);

Here is an Example for this,

alter table Student add(address char); The above command will add a new column *address* to the **Student** table

TO ADD MULTIPLE COLUMN TO EXISTING TABLE

- Using alter command we can even add multiple columns to an existing table. Following is the Syntax,

alter table *table-name* add(**column-name1** *datatype1*, **column-name2** *datatype2*, **column-name3** *datatype3*); Here is an Example for this,

alter table Student add(father-name varchar(60), mother-name varchar(60), dob date); The above command will add three new columns to the **Student** table

TO ADD COLUMN WITH DEFAULT VALUE

- alter command can add a new column to an existing table with default values. Following is the Syntax,

alter table *table-name* add(**column-name1** *datatype1* **default** *data*); Here is an Example for this,

- alter table Student add(dob date default '1-Jan-99'); The above command will add a new column with default value to the **Student** table

TO MODIFY AN EXISTING COLUMN

- alter command is used to modify data type of an existing column . Following is the Syntax,

alter table *table-name* modify(**column-name** *datatype*); Here is an Example for this,

- alter table Student modify(address varchar(30)); The above command will modify *address* column of the **Student table**

TO RENAME A COLUMN

- Using alter command you can rename an existing column. Following is the Syntax,

alter table table-*name* **rename** old-column-name to column-name; Here is an Example for this,

- alter table Student rename address to Location; The above command will rename *address* column to *Location*.

TO DROP A COLUMN

- alter command is also used to drop columns also. Following is the Syntax,

alter table *table-name* drop(column-name);

Here is an Example for this,

- alter table Student drop(address); The above command will drop *address* column from the **Student table**

TRUNCATE COMMAND

- *truncate* command removes all records from a table. But this command will not destroy the table's structure. When we apply truncate command on a table its Primary key is initialized. Following is its Syntax,

truncate table *table-name*

Here is an Example explaining it.

- truncate table Student; The above query will delete all the records of **Student** table.

DIFFERENCE BETWEEN DELETE AND TRUNCATE COMMAND

- **truncate** command is different from **delete** command. delete command will delete all the rows from a table whereas truncate command re-initializes a table(like a newly created table).
- **For eg.** If you have a table with 10 rows and an auto_increment primary key, if you use *delete* command to delete all the rows, it will delete all the rows, but will not initialize the primary key, hence if you will insert any row after using delete command, the auto_increment primary key will start from 11. But in case of *truncate* command, primary key is re-initialized.

DROP COMMAND

- *drop* query completely removes a table from database. This command will also destroy the table structure. Following is its Syntax,

drop table *table-name*

Here is an Example explaining it.

- drop table Student; The above query will delete the **Student** table completely. It can also be used on Databases. For Example, to drop a database,
- drop database Test; The above query will drop a database named **Test** from the system.

RENAME QUERY

- *rename* command is used to rename a table. Following is its Syntax,

rename table *old-table-name* to *new-table-name*

Here is an Example explaining it.

- rename table Student to Student-record;
- The above query will rename **Student** table to **Student-record**.

DML COMMAND

- Data Manipulation Language (DML) statements are used for managing data in database.

- DML commands are not auto-committed. It means changes made by DML command are not permanent to database,

- it can be rolled back.

INSERT COMMAND

- Insert command is used to insert data into a table.

Following is its general syntax,

INSERT into *table-name* values(data1,data2,..)

Lets see an example,

Consider a table **Student** with following fields.

- S_id S_Name age

INSERT into Student values(101,'Adam',15);

The above command will insert a record into **Student** table.

S_id	S_Name	age
101	Adam	

EXAMPLE TO INSERT NULL VALUE TO A COLUMN

- Both the statements below will insert NULL value into **age** column of the Student table.

```
INSERT into Student(id,name) values(102,'Alex');
```

Or,

```
INSERT into Student values(102,'Alex',null);
```

The above command will insert only two column value other column is set to null.

S_id	S_Name	age
101	Adam	15
102	Alex	

EXAMPLE TO INSERT DEFAULT VALUE TO A COLUMN

- INSERT into Student values(103,'Chris',default)

S_id	S_Name	age
101	Adam	15
102	Alex	
103	chris	14

Suppose the **age** column of student table has default value of 14.

Also, if you run the below query, it will insert default value into the age column, whatever the default value may be.

- INSERT into Student values(103,'Chris')

UPDATE COMMAND

- Update command is used to update a row of a table. Following is its general syntax,
  - **UPDATE** table-*name* set column-name = value *where* **condition**;

Lets see an example,

update Student set age=18 where s_id=102;

S_Id	S_Name	Age
101	Adam	15
102	Alex	18
103	Chris	14

EXAMPLE TO UPDATE MULTIPLE COLUMNS

- UPDATE Student set s_name='Abhi',age=17 where s_id=103;

The above command will update two columns of a record.

S_id	S_Name	Age
101	Adam	15
102	Alex	18
103	Abhi	17

DELETE COMMAND

- Delete command is used to delete data from a table. Delete command can also be used with condition to delete a particular row. Following is its general syntax,

DELETE from *table-name*;

Example to Delete all Records from a Table

```
DELETE from Student;
```

- The above command will delete all the records from **Student** table.

EXAMPLE TO DELETE A PARTICULAR RECORD FROM A TABLE

- Consider the following **Student** table

```
DELETE from Student where s_id=103;
```

S_id	S_Name	Age
101	Adam	15
102	Alex	18
103	Abhi	17

CONTINUE..

- The above command will delete the record where s_id is 103 from **Student** table.

S_id	S_name	Age
101	Adam	15
102	Alex	18

DCL COMMANDS

- Data Control Language(DCL) is used to control privilege in Database. To perform any operation in the database, such as for creating tables, sequences or views we need privileges.

Privileges are of two types,

- **System :** creating session, table etc are all types of system privilege.
- **Object :** any command or query to work on tables comes under object privilege.

DCL defines two commands,

- **Grant :** Gives user access privileges to database.
- **Revoke :** Take back permissions from user.

CONTINUED..

- **To Allow a User to create Session**

grant create session to *username*;

- **To Allow a User to create Table**

grant create table to *username*;

To provide User with some Space on Tablespace to store Table

alter user *username* quota unlimited on system;

To Grant all privilege to a User

- **grant** sysdba to *username*

CONTINUED..

- **To Grant permission to Create any Table**

  grant *create* any table to *username*

To Grant permission to Drop any Table

  grant *drop* any table to *username*

- **To take back Permissions**

  revoke create table from *username*

TCL COMMAND

Transaction Control Language(TCL) commands are used to manage transactions in database.

These are used to manage the changes made by DML statements.

It also allows statements to be grouped together into logical transactions.

CONTINUED..

Commit command

Commit command is used to permanently save any transaction into database.

Following is Commit command's syntax,

commit;

Rollback command

This command restores the database to last commited state. It is also use with savepoint command to jump to a savepoint in a transaction.

Following is Rollback command's syntax,

rollback to *savepoint-name*;

Savepoint command

- **savepoint** command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

Following is savepoint command's syntax,

savepoint *savepoint-name*;

MCQ: EXPECTED QUESTIONS

Every time attribute A appears, it is matched with the same value of attribute B, but not the same value of attribute C. Therefore, it is true that: (IBPS IT Officer Exam 2013 Morning Shift)

A. $A \rightarrow B$.

B. $A \rightarrow C$.

C. $A \rightarrow (B,C)$.

D. $(B,C) \rightarrow A$.

SOLUTION OF Q.1

Option **A** A $\longrightarrow$ B.

Q. 2

The different classes of relations created by the technique for preventing modification anomalies are called:

(IBPS IT Officer Exam 2013 Morning Shift)

A. normal forms.

B. referential integrity constraints.

C. functional dependencies

D. None of the above is correct.

SOLUTION OF Q.2

Option **A** normal forms.

Q.3

Row is synonymous with the term:

(IBPS IT Officer Exam 2013 Morning Shift)

A. record.

B. relation.

C. column.

D. field.

SOLUTION OF Q.3

Option **A Record**

Q.4

The primary key is selected from the:

(IBPS IT Officer Exam 2013 Morning Shift)

- A. composite keys.
- B. determinants.
- C. candidate keys.
- D. foreign keys.

SOLUTION OF Q.4

Option C. candidate keys.

Q.5

Which of the following is a group of one or more attributes that uniquely identifies a row?

(IBPS IT Officer Exam 2013 Morning Shift)

- A. Key
- B. Determinant
- C. Tuple
- D. Relation

Solution of Q.5

Option A. Key

Q. 6

- When the values in one or more attributes being used as a foreign key must exist in another set of one or more attributes in another table, we have created a(n):

(SBI IT Officer Exam 2011)

A. transitive dependency.

B. insertion anomaly.

C. referential integrity constraint.

D. normal form.

SOLUTION OF Q.6

Option C. referential integrity constraint.

Q.7

A functional dependency is a relationship between or among: (SBI IT Officer Exam 2011)

A. Tables.

B. rows.

C. relations.

D. attributes.

SOLUTION OF Q.7

Option D. attributes.

Q. 8

Which of the following is not a restriction for a table to be a relation?

(SBI IT Officer Exam 2011)

A. The cells of the table must contain a single value.
B. All of the entries in any column must be of the same kind.
C. The columns must be ordered.
D. No two rows in a table may be identical.

SOLUTION OF Q.8

Option c. The columns must be ordered.

Q. 9

In the ___________ normal form, a composite attribute is converted to individual attributes.

A) First
B) Second
C) Third
D) Fourth

SOLUTION OF Q.9

Answer: a

Explanation: The first normal form is used to eliminate the duplicate information.

Q. 10

Which one of the following is a set of one or more attributes taken collectively to uniquely identify a record ?

a) Candidate key
b) Sub key
c) Super key
d) Foreign key

SOLUTION OF Q.10

Answer:c

Explanation: Super key is the superset of all the keys in a relation.

Thank You
shift