UNIT-II - Introduction to Relational Model and SQL:

Introduction to Relational model - structure of Relational Database - Relational Database design - Keys - Database schema and schema Diagrams - Relational Query Language : overview of the SQL Query Language - SQL Constraints :.

2.1 (i) Relation Model:-

The relational model was first introduced by Ted Codd of IBM Research in 1970 in a classic paper.

The relational Model for database Management is an approach to logically represent and manage the data stored in a data base.

→ RM represents the database as a collection of relations. A relation is nothing but a table of values.

→ The data is organized into a collection of two dimensional inter-related tables, also known as relations.

→ Each relation is a collection of columns and rows.

→ Column represents the attributes of an entity and the row (or tuples) represents the records.

Ex: Consider a relation student with attributes ROLL NO, NAME, ADDRESS, PHONE & AGE.

→ R.N	NAME	ADD	PHONE	AGE
1.	RAM	chennai	12345678	18
2.	RANESH	Vellore	87623521	19
3.	KUMAR	Srinagar	12345769	20

Primary Key

Tuple (rows)

Columns (Attributes)

As we can notice from the above relation:

→ Any given row of the relation indicates a student i.e., the row of the tables describes a real-world entity.

→ The columns of the table indicate the attributes related to the entity. In this case, the rollno, CGPA, & the name of the student.

Highlights:

1. Relational model stores the data into tables [relations].
2. It makes data sorting and data access easier.
3. Provides a standard way to organize data in database.

Relational Model Concepts:-

The relational model represents the database as a collection of relations. Each relation resembles a table of values or to some extent, a flat file of records.

When a relation is thought of as a table of values, each row in the table represents a collection of related data values.

In the relational model, each row in the table represents a fact that typically corresponds to a real-world entity (or) relationship.

The table name & column names are used to help to interpret the meaning of the values in each row.

Ex: STUDENT DETAILS:

student

name	student_num	class	major

Course:

Cou_name	Cou_NO	Credit_hrs	Department

section:

Section Id	Cou_Numb	Sem	Year	Instructor

Grade Report:

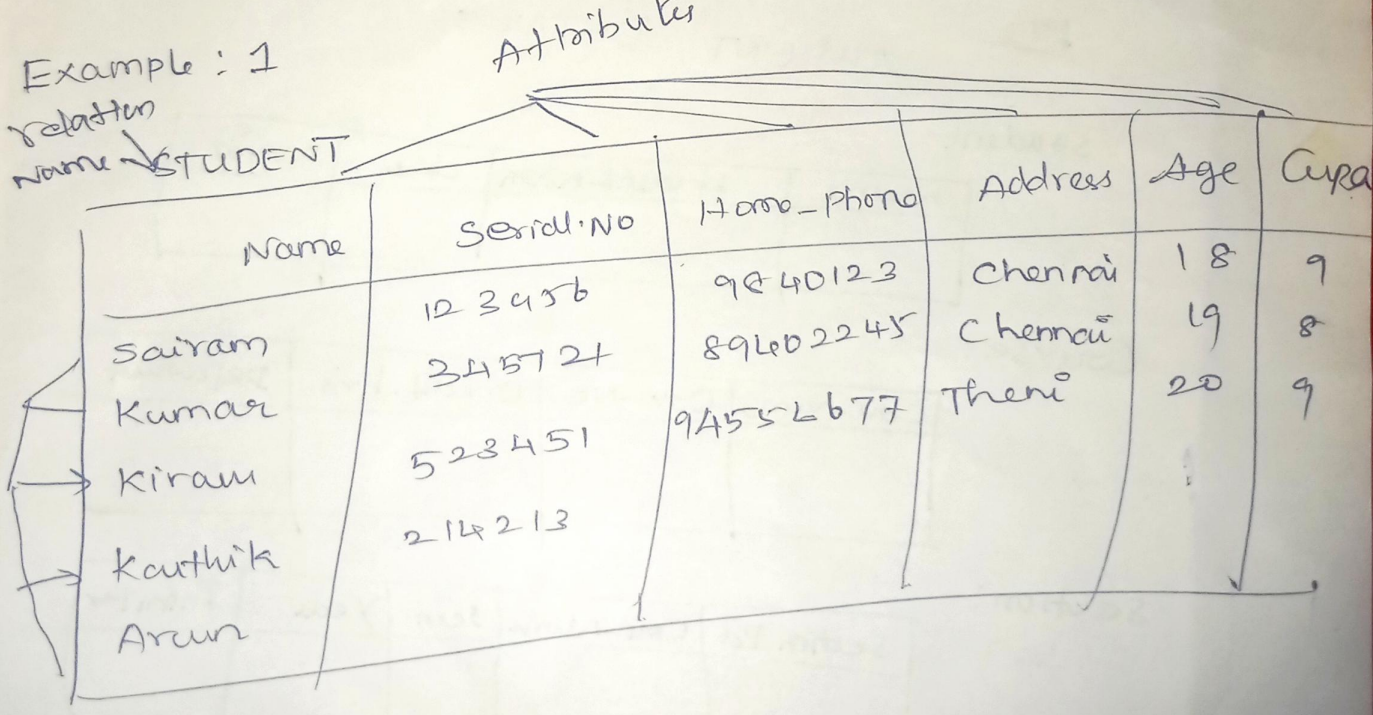
StdNum	Section	Grade

Each row represents facts about a particular student entity. The column names - Name, Stud.No, Class & Major - specify how to interpret the data values in each row, based on the column each value. All values in a column are of the same data type.

In the formal relational model terminology a row called a tuple. a column header is called an attribute. and the table is called a relation.

The data type describing the types of values that can appear in each column is represented by a domain of possible values.

Example : 1



2.2 Structure of Relational Databases:

A relational database consists of a collection of tables, each of which assigned a unique name.

Relational Data Structure:

1) **Attribute**: Each column in a Table. Attributes are the properties which define a relation.

   Ex: student_Name, stu_Ser.No, etc...

2) **Tables**: In the relational data model, the relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns.
   - → Rows represent records
   - → Columns represent attributes

3) **Tuple**: It is nothing but a single row of a table which contains a single record.

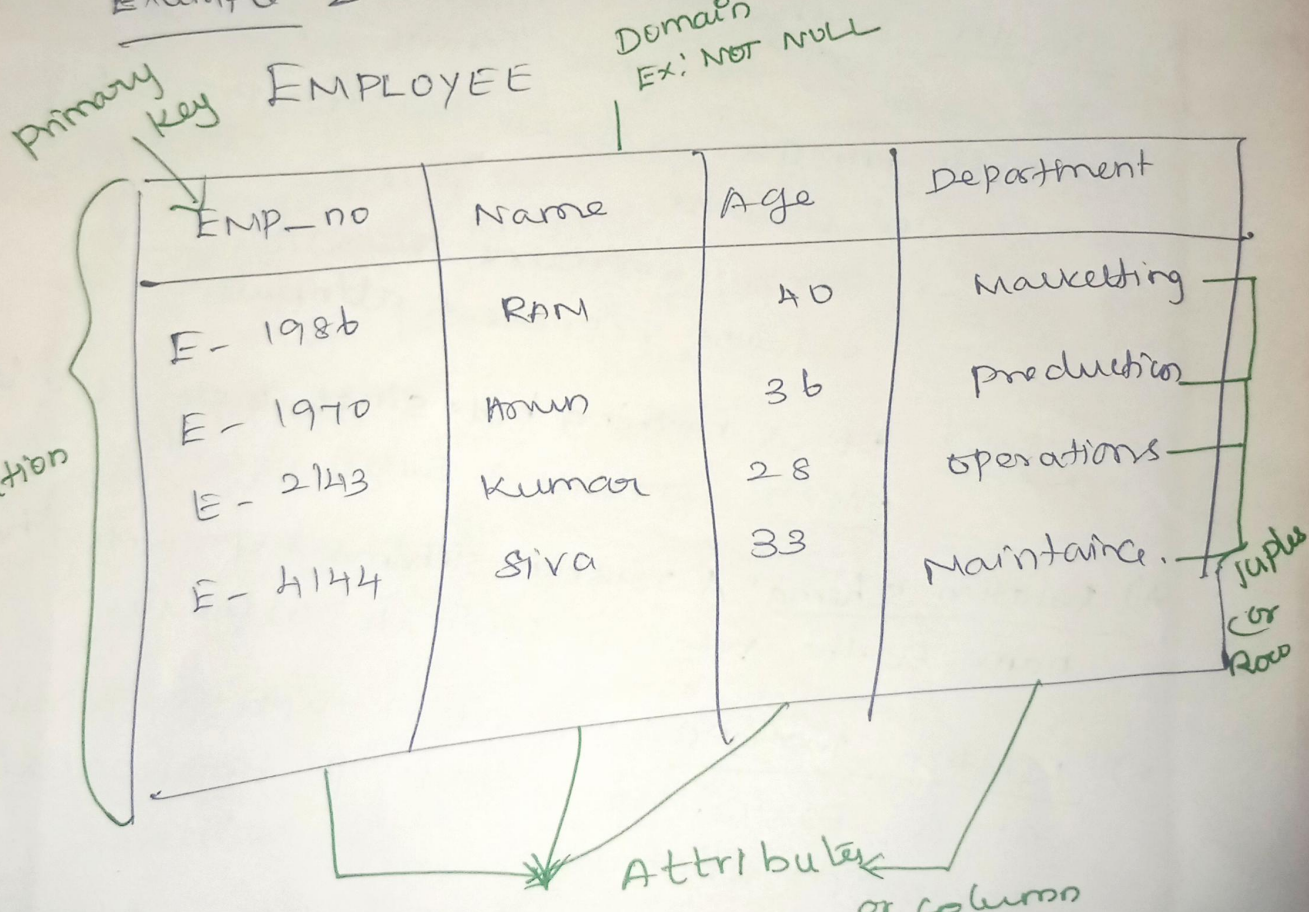
4) **Relation Schema**: A relation schema represents the name of the relation with its attributes

5) **Relation Instance**: The set of tuples of a relation at a particular instance of time is called as relation Instance.

6) **Attribute Domain**: Set of Pre-defined atomic value that an attribute can take i.e., it describes the legal values that an attribute can take. (pool of value is Domain)

Formal relational term	Informal equivalence
Tuple	row or record
Attribute	Column (or) field
relation	table
domain	Pool of legal values
Key	unique identifier

Example' 2



Example: 3

Any table can be shown as a structure of relation.

Data Integrity:

Integrity constraints provide a means of ensuring that changes made to the database by authorized users do not result in a loss of data consistency.

Types:

1. Domain Constraints
2. Referential integrity
3. Nulls
4. Entity integrity.

Domain Constraints: Dc specify the set of values that can be associated with an attribute. they are easily tested by the system whenever a new data item is entered into the database.

Referential Integrity: A value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.

NULLS: NULL Register it represents a value for an attribute that is currently unknown or is not applicable for this tuple.

Entity Integrity: In base relation no attribute of Primary key can be NULL. A primary key is used to identify the tuple uniquely.

Enterprise Constraints: These are additional rules specified by the users or administrators.

Data Manipulation:-

The manipulative part of relational model consists of a set of operators known collectively as the relational algebra together with relation calculus.

Advantage of Relational Model:

→ **Simplicity:** A relational data model is simpler than the hierarchical & N/w model.

→ **Structural Independence:** The relational database is only concerned with data & not with a structure. This can improve the performance of the model.

→ **Easy to Use:** The RM is easy as tables consisting of rows & columns is quite natural and simple to understand.

→ **Query Capability:** It makes possible for a high-level query Language like SQL to avoid complex database.

→ **Data independence:** The structure of a database can be changed without having to change any application.

→ **Scalable:** Regarding a no. of records, (or) rows & no. of fields, a database should be enlarged to enhance its usability.

Disadvantage:

→ Few relational DB have limits on field Lengths which cannot be exceeded.

→ Relational databases can sometimes become complex as the amount of data grows.

→ Complex relational database system may lead to isolated databases where the information cannot be shared from one system to another.

2.3
Rational Database Design:

Relational database Design models information and data into a set of tables with rows and columns.

Each row of a relation/table represents a record, and each column represents an attribute of data.

The Structured Query Language (SQL) is used to manipulate relational database. The design of a relational data is 4 stages.

1) Define relations/attributes: The various tables and attributes related to each table are identified. The tables represent entities, and the attributes represent the properties of the respective entities.

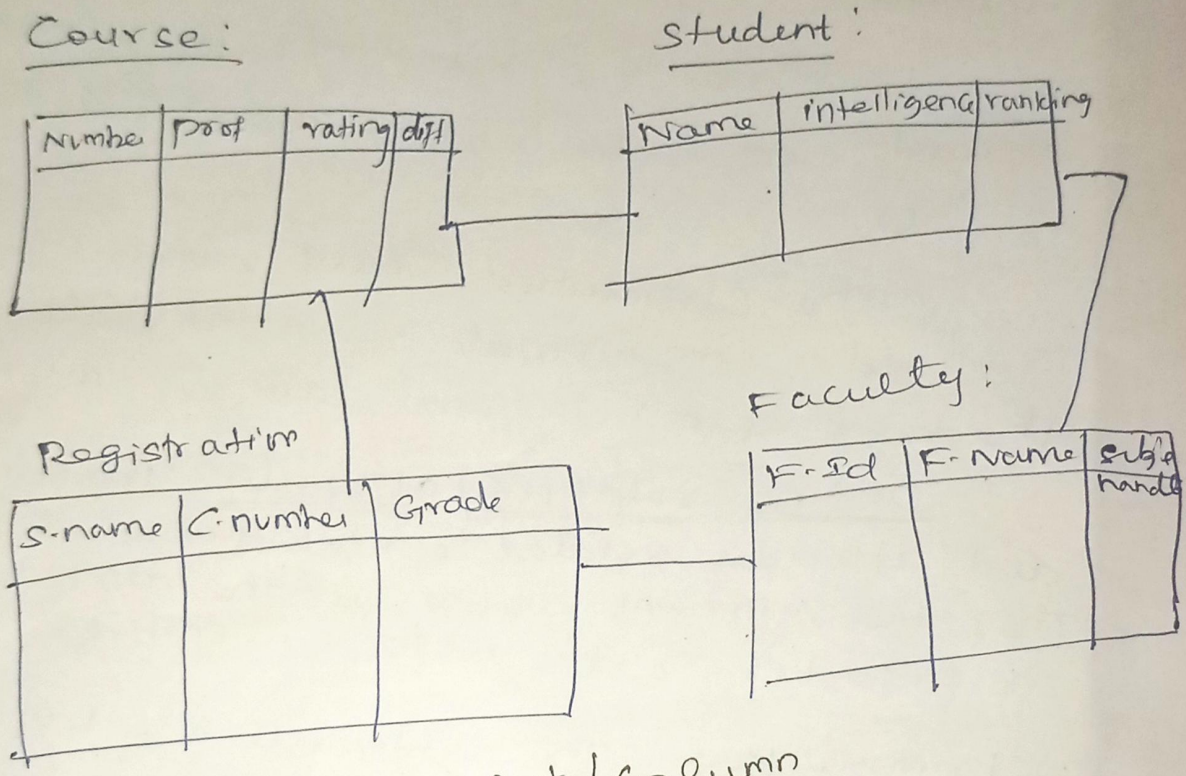
2) Primary keys: The attribute or set of attributes that help in uniquely identifying a record is identified and assigned as the primary key.

3) Relationship: The relationships b/w the various tables are established with the help of foreign keys. Foreign keys are attributes occuring in a table that are primary keys of another table.

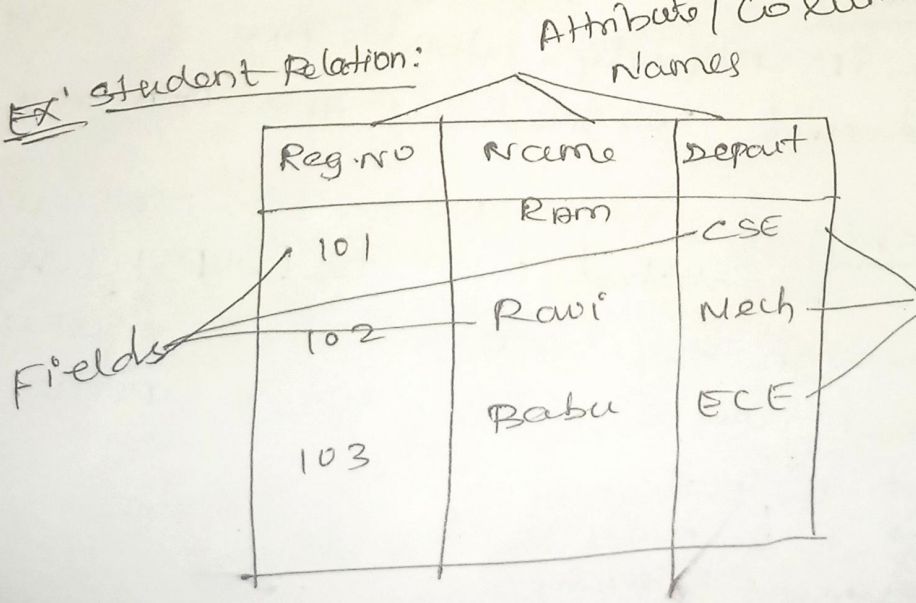
- → 1 to 1
- → 1 to many
- → many to many

4) Normalization:- This is the process of optimizing the database structure. Normalization simplifies the database design to avoid redundancy and confusion.

Relational Database Design Example:



Ex: Student Relation:



Student Relation in Relational Model.

2.1

Keys:

A DBIS key is an attribute or set of an attribute which helps you to identify a row(tuple) in a relation [Model].

→ They allow you to find the relation between two tables.

→ Key helps you uniquely identify a row in a table by a combination of one (or more) columns in that table.

Ex:

Stu-id	Stu-name	Email-ID

In the above given example. Stu-ID is a Primary key because it uniquely identifies a Student record.

Why we need a Key?

→ Keys help you to identify any row of data in a table.

→ Keys allow you to establish a relationship between and identify the relation b/w tables.

→ Key helps you to enforce identity in the relationship.

Various keys in DBMS:

1. Super key
2. Primary key
3. Candidate key
4. Alternate key
5. Foreign key

1) Superkey:

→ super key is a set of an attribute which can uniquely identify a tuple.

→ superkey is a superset of a candidate key.

Ex:

STUD-ID	STUD-NAME	Reg.NO	EMAIL ID

Ex:

→ In the above STUDENT table, for (STUD-ID, STUD-NAME) the name of two students can be the same, but their STUD-ID can't the same.

→ The super key would be STUD-ID, (STUD-ID, STUD-NAME) etc.

=) Primary Key:

A column or group of columns in a table which helps us to uniquely identify every row in the table called a Primary key.

→ The same value can't appear more than once in the table.

Rules:

1. The primary key field cannot be null.
2. The value in a primary key column can never be modified (or) updated if foreign key refers to that primary key.

STUD-ID	STUD-NAME	Email-ID

→ From the above example, STUD-ID is a primary key.

3) 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.

STUD-ID	STUD-NAME	Email-ID	RgNO

→ In the STUDENT table, STUD-ID is best suited primary key.

→ Rest of the attributes like Reg.NO, Email-ID are considered as candidate keys. It helps us to uniquely identify the st-record table.

Alternate Key:

→ Alternate keys is a column or group of column in a table that uniquely identify every row in that table.

→ A table can have multiple choice for a primary key but only one can be set as the primary key.

→ All the keys which are not primary key are called Alternate key.

Ex: In this table, STUD_ID, REN_NO, EMAIL_ID are qualified to become a primary key.

STUD_ID	REN_NO	STUD_NAME	Email_ID

Suppose, since STUD_ID is the primary key, REN_NO, EMAIL_ID becomes the alternate key.

Foreign key:

→ Foreign key is a column that creates a relationship between two tables.

→ The purposes of foreign keys is to maintain data integrity and allow navigation b/w two different instance of an entity

→ It acts as a cross-reference between two table as it refers the primary key of another table.

Table Name: Faculty

Foreign key.

Fac_ID	Fac_Name	Dep_Code	Email_Id

Table Name: Department

Dept_Code	Dept_Name

Dept_code is the foreign key.

Database Schema and schema Diagrams

Schema

A Schema can be defined as the design of a database

database schema

→ The overall description of the database is called database schema

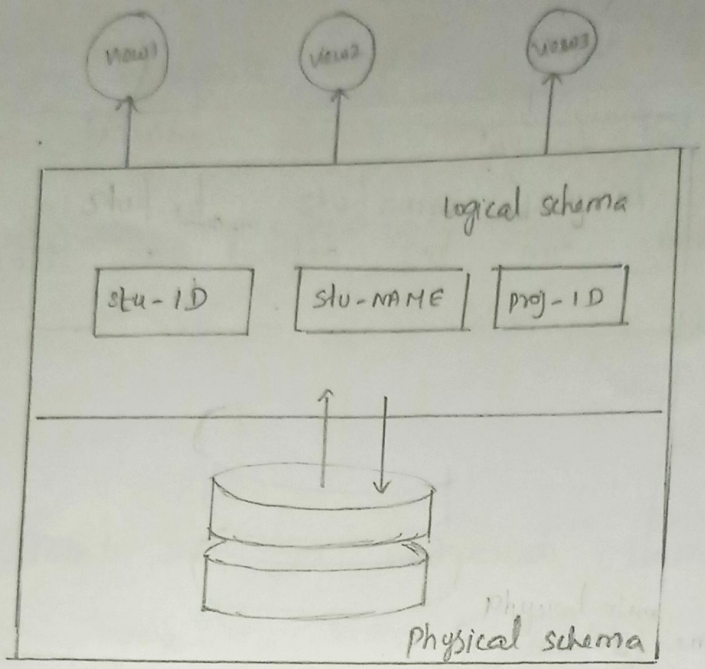
→ A database schema is the skeleton structure that represents the logical view of the entire database

→ It defines how the data is organized and how the relations among them are associated

→ A database schema does not contain any data or Information

→ A database schema can be divided broadly into two categories.

1. Physical Database schema
2. logical Database schema



Physical Database Schema :-

This schema pertains to the actual storage of data and its form of storage like files, indices etc. It defines how the data will be stored in a secondary storage.

Logical Database Schema :

This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views and integrity constraints

Schema objects

→ Database has objects like tables, views, indexes, constraints, procedures, packages etc.

→ All these objects are owned by particular user. This user is called as schema

→ Multiple schemas with its own objects

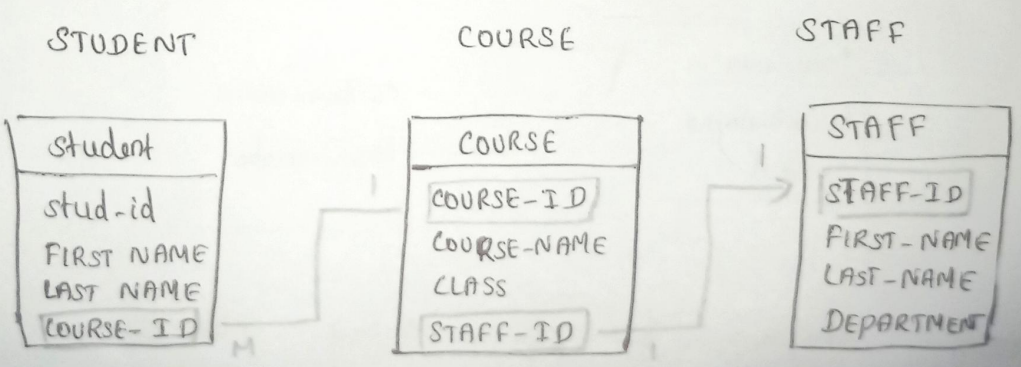
→ Each objects in one schema can be different from other schema objects

→ Each schema have specific role and purpose for task

→ one schema cannot access object of another schema.

Example:

Consider three schemas :- STUDENT, COURSE, STAFF



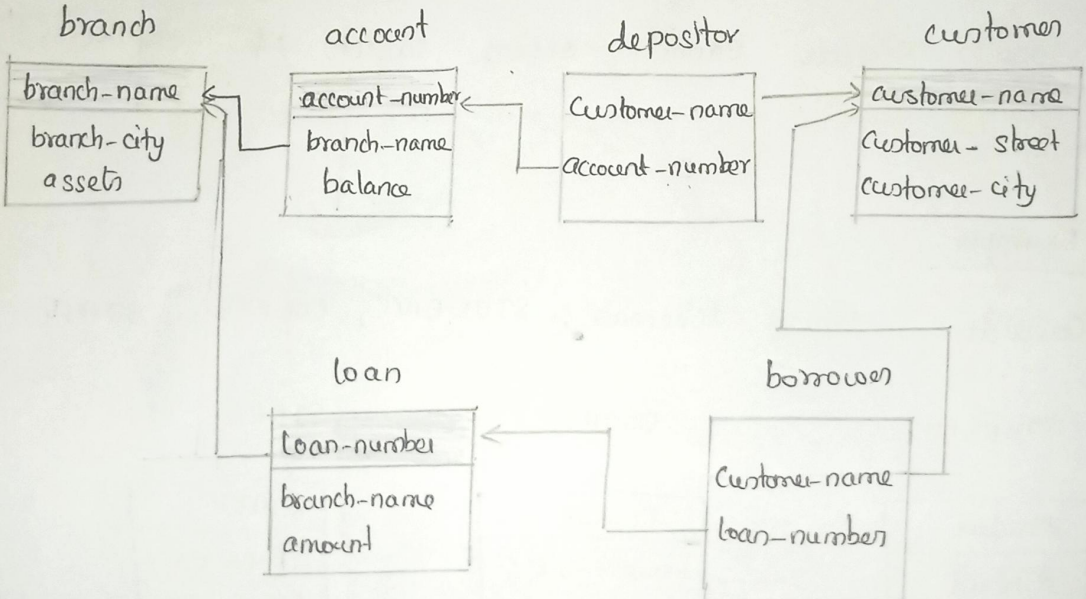
INSTANCE.

→ Schema contains structure and mapping of the object in it

→ It records values in the schema objects keeps changing and those structures with their value at any point in time is called instance of the schema.

Schema Diagrams:-

A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams.



Schema diagram for the banking enterprise.

-> It is customary to list the primary key attributes of a relation schema before the other attributes.

-> for example, the branch-name attribute of branch-schema is listed first, since it is the primary key

-> The above diagram shows for our banking enterprise

-> Each relation appears as a box, with the attributes listed ~~inside~~ inside it and the relation name above it

2.6 Relational Query languages.

→ A query language is a language in which a user requests information from the database

→ These languages are usually on a level higher than that of a standard programming language

→ Query languages can be categorized as either procedural or non-procedural

Procedural language:-

In a procedural language, the user instruct the system to perform a sequence of operations on the database to compute the desired result.

Non-procedural language:-

In a Non-procedural language, the user describes the desired information without giving a specific procedure for obtaining that information.

Overview of the SQL Query language:

IBM developed the original version of SQL, originally called Sequel. The Sequel language has evolved since then, and its name has changed to SQL (Structured Query language). SQL is a computer language aim to store, manipulate and query data stored in relational db

→ The SQL language has several Parts

* Data-definition language (DDL).

The SQL DDL provides commands for defining relation schemas, deleting relations and modifying relation schemas.

* Data-manipulation language (DML)

The SQL DML provides the ability to query information from the database and to insert tuples into, delete tuples from and modify tuples in the database.

* Integrity:

The SQL DDL includes commands for specifying integrity constraints that the data stored

in the database must satisfy.

View definition:-

The SQL DDL includes commands for defining views

Transaction control:-

SQL includes commands for specifying the beginning and ending of transactions

Embedded SQL and dynamic SQL:

Embedded and dynamic SQL define how SQL statements can be embedded within general-purpose-programming languages such as C, C++ and Java

Authorization:-

The SQL DDL includes commands for specifying access rights to relations and views.

SQL Data Definition.

The set of relations in a database must be specified to the system by means of a data-definition language (DDL)

Basic types:-

The SQL standard supports a variety of built-in-types, including

char (n):-

A fixed length character string with user specified length n. The full form, character can be used instead.

varchar (n)

A variable-length character string with user-specified maximum length n. The full form character varying is equivalent

int:-

An integer. (a finite set of the integers that is machine dependent

Smallint :-

A small integer (a machine-dependent subset of the Integer Type)

numeric(p,d) :-

→ A fixed-point number with user-specified Precision.

→ The number consist of p digit and d of the p digits are to the right of the decimal point.

real double precision :-

Floating-point and double precision floating-point numbers with machine-dependent precision.

float(n) :-

A floating point number, with precision of at least n digits

Data Definition Language (DDL)

A data definition language (DDL) statement are used to define the database structure or schema

DDL commands

CREATE

ALTER

DROP

RENAME

TRUNCATE

Create Table

To make a new database, table, index or stored query. A create statement in SQL creates an object inside of a relational database management system (RDBMS)

Syntax:-

```
CREATE TABLE table_name
(
   column_name1  data_type,
   column_name2  data_type ...
   column_namen  data type

);
```

Example:

```
create table employee
(
empid number (10) primary key,
```

empname varchar2(20) not null,

dob date,

doj date,

dept char(10),

salary number(7)

);

ALTER A TABLE

To modify an existing database object. Alter the Structure of the database

a. To add a column in a table

Syntax: ALTER TABLE table-name ADD column-name datatype;

Example: SQL> alter table employee add exp number(2).

b. To delete a column in a table

Syntax: ALTER TABLE table-name DROP column column-name.

Example: SQL > alter table employee drop column exp.

c. To change the data type of a column in a table

Syntax: ALTER TABLE table-name MODIFY column-name datatype.

Example: SQL > alter table employee modify empname varchar2(30)

d. To rename a column in a table

Syntax: ALTER TABLE tablename RENAME COLUMN

old-column TO new-column;

Example: SQL > alter table employee rename column empname to ename;

Table altered.

RENAME table

Purpose: Rename an object

Syntax: RENAME old_table_name to new_table_name;

Example: rename employee to emp;

DROP Table:

Delete objects from database

Syntax: DROP TABLE table_name;

Example: drop table employee;

TRUNCATE Table:

Remove all records from a table, including all spaces allocated for the records are removed.

Syntax: TRUNCATE TABLE table_name;

Example: truncate table employee;

Data Manipulation language. (DML)

-> Data Manipulation language allows the user to query and manipulate data in existing schema in object

-> It allows following data to insert, delete, update and recovery data in schema object

DML Commands

1. INSERT
2. UPDATE
3. DELETE
4. SELECT

INSERT

Values can be inserted into table using Insert command

Syntax :-

```
INSERT INTO table-name VALUES (value1, value2, ... );
```

example:

```
Insert into employee values (10144, 'Bharath', 12-April-2000',
                             25-April-2019, 61000).
```

8. UPDATE

This command is used for updating or modifying the values of column in a table

Syntax:-

UPDATE table_name SET Colum_name = new value [WHERE condition];

Example:-

update employee set empid = '10159' where empname = 'Bharath';

DELETE

This command is used for removing one or more record from a table

Syntax: DELETE FROM table_name [WHERE Condition];

Example: delete from employee where empid = '1009';

SELECT

Retreive (read) data from a database

Syntax: SELECT Column_name(s) FROM table_name;

example: SQL> select empid, empname from employee;

EMPID	EMPNAME
10159	Bhaiath
10166	Sriram
10193	Hasini
10101	Hasinath
10240	Krishna

DATA CONTROL LANGUAGE (DCL)

DCL include commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system

CREATE

GRANT

REVOKE

CREATE:

Create the user area in the database

Syntax: CREATE USER username IDENTIFIED BY password

Example: SQL > create user bihen identified by cse123;

GRANT:

Gives user's access privileges (Rights) to database

Syntax: GRANT ALL PRIVILEGES TO Username;

Example: SQL > grant all privileges to biher;

Grant succeeded

SQL > connect

Enter user-name : biher

Enter password :

connected

REVOKE

withdraw user's access privileges given by using the GRANT command

Syntax: REVOKE ALL PRIVILEGES FROM Username;

Example: SQL> revoke all privileges from biher;

Revoke succeeded

SQL > connect

Enter user-name : biher

Enter password

ERROR

ORA - 01045: user BIHER lacks CREATE SESSION Privilege, logon denied

Syntax: GRANT ALL PRIVILEGES TO username;

Example: SQL > grant all privileges to biher;

Grant succeeded

SQL > connect

Enter user-name : biher

Enter password :

connected

REVOKE.

withdraw user's access privileges given by using the GRANT command

Syntax: REVOKE ALL PRIVILEGES FROM username;

Example: SQL> revoke all privileges from biher;

Revoke succeeded

SQL > connect

Enter user-name : biher

Enter password

ERROR

ORA - 01045: user BIHER lacks CREATE SESSION Privilege, logon denied

Warning: You are no longer connected to ORACLE

TRANSACTION CONTROL LANGUAGE (TCL)

These commands are used to manage transactions in the database.

TCL Commands

COMMIT

ROLLBACK

SAVEPOINT

COMMIT

-> Commits a transaction

-> used to permanently save any transaction in the db

Syntax: COMMIT;

ROLLBACK:

Rollbacks a transaction in case of any error occurs

Syntax: ROLLBACK;

SAVEPOINT:

Sets a savepoint within a transaction

syntax: SAVEPOINTS savepointname;

2.8 SQL constraints

It is a mechanism used to present invalid data entry into the table

Types:-

(i) Domain Integrity constraints

(ii) Entity Integrity constraints

(iii) Referential integrity constraints

(i) Domain Integrity Constraints

-> Each table has certain set of columns and each column allows a same type of data, based on its data type

-> The column does not accept values of any other data type

a) NOT NULL constraint

-> By default, a column can hold NULL

-> If you do not want to allow NULL value in a column, you can place the NOT NULL constraint on this column.

Example:-

```
CREATE TABLE Student
(
   Regno    Number (10)   not null,
   Stuname  Varchar (25)  not null,
   Age      Number (3),
   Dept     char (10)
);
```

6) CHECK Constraint

The check constraint ensures that all values in a column satisfy certain conditions

```
create table boyshostel
(
   Hostelid   Number (10)   not null,
   Regno      Number (10)   not null,
   stuname    Varchar (25)  not null,
   Gender     Char (8)      CHECK (gender = 'male')
);
```

ii) Entity Integrity constraints:-

It is concerned with ensuring that each row of a table has a unique and non-null value

-> There are two types of entity-Integrity constraints

1. UNIQUE
2. PRIMARY KEY

UNIQUE:

Unique constraints is used to prevent duplication of values and null values.

Example:

```
create table boyshostel
(
Hostelid    number(10)    unique,
Regno       Number(10)    notnull,
Stuname     varchar(25)   notnull,
Gender      char(8)       check (gender='male')
);
```

b) PRIMARY KEY constraint

This is used for to prevent duplication of values and null values

Example:-

```
create table boyshostel
(
Hostelid    Number(10)    primary key,
Regno       number(10)    notnull,
stuname     varchar(25)   notnull,
Gender      char(8)       check (gender='male');
);
```

(iii) Referential Integrity Constraint

→ To establish a parent child relationship between two tables having a common column we can use referential integrity constraints.

→ To Implement this, we should define the column in the parent table as primary key and the same column in the child table as a foreign key referring to the corresponding parent entry

Example:

Parent table

```
create table student
(
  Regno    Number(10)   primarykey,
  Stuname  varchar(25)  not null,
  Dept     char(10),
  Age      Number(3),
  Gender   char
);
```

child table

```
create table boyshostel
(
  Hostelid   Number(10)   primarykey,
  Regno      Number(10)   references student(Regno),
  Stuname    varchar(25)  not null,
  Gender     char(8)      check(gender='male'));
```