BE23CS405 - DATABASE MANAGEMENT SYSTEMS

UNIT-1

Purpose of Database System – Views of data - Data Models – Database System Architecture – Introduction to relational databases – Relational Model – Constraints – Relational Algebra. Overview of the SQL Query Language – Basic Structure of SQL Queries – DDL – DML – Keys.

Introduction

• **Definition:** A Database Management System (DBMS) is a collection of interrelated data and various programs that are used to handle that data.

• The primary goal of DBMS is to provide a way to store and retrieve the required information from the database in convenient and efficient manner.

• For managing the data in the database two important tasks are conducted -

(i) Define the structure for storage of information.

(ii) Provide mechanism for manipulation of information.

• In addition, the database systems must ensure the safety of information stored.

Database System Applications

There are wide range of applications that make use of database systems. Some of the applications are -

1) Accounting: Database systems are used in maintaining information employees, salaries, and payroll taxes.

2) Manufacturing: For management of supply chain and tracking production of items in factories database systems are maintained.

3) For maintaining customer, product and purchase information the databases are used.

4) Banking: In banking sector, for customer information, accounts and loan and for performing banking applications the DBMS is used.

5) For purchase on credit cards and generation of monthly statements database systems are useful.

6) Universities: The database systems are used in universities for maintaining student information, course registration, and accounting.

7) Reservation systems: In airline/railway reservation systems, the database is used to at maintain the reservation and schedule information.

8) Telecommunication: In telecommunications for keeping records of the calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about communication networks the database systems are used.

Purpose of Database System

AU: May-07, 12, Dec.-04, Marks 8

• Earlier database systems are created in response to manage the commercial data. These data is typically stored in files. To allow users to manipulate these files various programs are written for

1) Addition of new data

2) Updating the data

3) Deleting the data.

• As per the addition of new need, separate application programs were required to write. Thus as the time goes by, the system acquires more files and more application programs.

• This typical file processing system is supported by conventional operating system. Thus the file processing system can be described as -

• The system that stores the permanent records in files and it needs different application programs to extract or add the records.

Before introducing database management system, this file processing system was in use. However, such a system has many drawbacks. Let us discuss them

Disadvantages of Traditional File Processing System

The traditional file system has following disadvantages:

1) Data redundancy: Data redundancy means duplication of data at several places. Since different programmers create different files and these files might have different structures, there are chances that some information may appear repeatedly in some or more format at several places.

2) Data inconsistency: Data inconsistency occurs when various copies of same data may no longer get matched. For example changed address of an employee may be reflected in one department and may not be available (or old address present) for other department.

3) Difficulty in accessing data: The conventional file system does not allow to retrieve the desired data in efficient and convenient manner.

4) Data isolation: As the data is scattered over several files and files may be in different formats, it becomes to retrieve the desired data from the file for writing the new application.

5) Integrity problems: Data integrity means data values entered in the database fall within a specified range and are of correct format. With the use of several files enforcing such constraint on the data becomes difficult.

6) Atomicity problems: An atomicity means particular operation must be carried out sib entirely or not at all with the database. It is difficult to ensure atomicity in conventional file processing system.

7) Concurrent access anomalies: For efficient execution, multiple users update data simultaneously, in such a case data need to be synchronized. As in traditional file systems, data is distributed over multiple files, one cannot access these files concurrently.

8) Security problems: Every user is not allowed to access all the data of database no system. Since application program in file system are added in an ad hoc manner, enforcing such security constraints become difficult.

Database systems offer solutions to all the above mentioned problems.

Difference between Database System and Conventional File System

Sr. No.	Database systems	Conventional file systems
1.	Data redundancy is **less.**	Data redundancy is **more.**
2.	**Security** is **high.**	Security is very **low.**
3.	Database systems are used when security **constraints are high.**	Conventional file systems are used where there is **less demand** for **security constraints.**
4.	Database systems define the data in a **structured** manner. Also there is well defined co-relation among the data.	File systems define the data in **un-structured** manner. Data is usually in isolated form.
5.	Data **inconsistency** is **less** in database systems.	Data **inconsistency** is **more** in file systems.
6.	**User** is **unknown** to the physical address of the data used in database systems.	User locates the **physical address of file** to access the data in conventional file systems.
7.	We can retrieve the data in any **desired format** using database systems.	We cannot retrieve the data in any desired format using file systems.
8.	There is ability to access the data **concurrently** using database systems.	There is **no ability to concurrently** access the data using conventional file system.

Characteristics of Database Systems

Following are the characteristics of database system

1) Representation of some aspects of real world applications.

2) Systematic management of information.

3) Representing the data by multiple views.

4) Efficient and easy implementation of various operations such as insertion, deletion and updation.

5) It maintains data for some specific purpose.

6) It represents logical relationship between records and data.

Advantages of Database Systems

Following are the advantages of DBMS -

1) DBMS removes the data redundancy that means there is no duplication of data in database, ends it

2) DBMS allows to retrieve the desired data in required format.

3) Data can be isolated in separate tables for convenient and efficient use.

4) Data can be accessed efficiently using a simple query language.

5) The data integrity can be maintained. That means - the constraints can be applied on data and it should be in some specific range.

6) The atomicity of data can be maintained. That means, if some operation is performed on one particular table of the database, then the change must be reflected for the entire database.

7) The DBMS allows concurrent access to multiple users by using the synchronization technique.

8) The security policies can be applied to DBMS to allow the user to access only desired part of the database system.

Disadvantages of Database Systems;

1) Complex design: Database design is complex, difficult and time consuming.

2) Hardware and software cost: Large amount of investment is needed to setup the required hardware or to repair software failure.

3) Damaged part: If one part of database is corrupted or damaged, then entire database may get affected.

4) Conversion cost: If the current system is in conventional file system and if we need to convert it to database systems then large amount of cost is incurred in purchasing different tools, and adopting different techniques as per the requirement.

5) Training: For designing and maintaining the database systems, the people need to be trained.

Review Questions

1. Compare file system with database system. ***AU: May-07, Marks 8, May-12, Marks 2***

2. *What are the advantages and disadvantages of DBMS?*

Views of Data

AU: May-16, Marks 16

• Database is a collection of interrelated data and set of programs that allow users to access or modify the data.

• Abstract view of the system is a view in which the system hides certain details of how the data are stored and maintained.

• The main purpose of database systems is to provide users with abstract view of the data.

• The view of the system helps the user to retrieve data efficiently.

• For simplifying the user interaction with the system there are several levels of abstraction - these levels are - Physical level, logical level and view level.

Data Abstraction

Data abstraction: Data abstraction means retrieving only required amount of information /of the system and hiding background details.

There are several levels of abstraction that simplify the user interactions with the system. These are

1) Physical level:

• This is the lowest level.

• This level describes how actually the data are stored.

• This level describes complex low level data structures.

2) Logical level:

• This is the next higher level, which describes the what data are stored in database.

• This level also describes the relationship among the data.

• The logical level thus describes then entire database in terms of small number of relatively simple structures.

• The database administrators use logical level of abstraction for deciding what information to keep in database.

3) View level:

This is highest level of abstraction that describes only part of the entire database.

• The view level can provide the access to only part of the database.

• This level helps in simplifying the interaction with the system.

• The system can provide multiple views of the same system.

• Clerk at the reservation system, can see only part of the database and can access the required information of the passenger.

Fig. 1.3.1 shows the relationship among the three levels of abstraction.

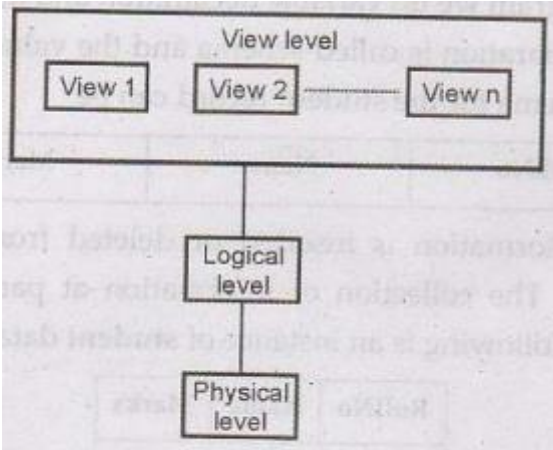


Fig. 1.3.1 : Levels of data abstraction

For example: Consider following record

Type employee = record

empID:numeric(10)

empname:char(20)

dept_no:numeric (10)

salary:numeric(8,2)

end

This code defines a new record employee with four fields. Each field is associated with field name and its type. There are several other records such as

department with fields dept_no, dept_name, building

customer with fields cust_id,cust_name

• At the physical level, the record - customer, employee, department can be Vibe described as block of consecutive storage locations. Many database systems hide lowest level storage details from database programmer.

• The type definition of the records is decided at the logical level. The programmer work of the record at this level, similarly database administrators also work at this level of abstraction.

• There is specific view of the record is allowed at the view level. For instance - - customer can view the name of the employee, or id of the employee but cannot access employee's salary.

Instances and Schemas

Schema: The overall design of the database is called schema

For example - In a program we do variable declaration and assignment of values to the variable. The variable declaration is called schema and the value assigned to the variable is called instance. The schema for the student record can be

RollNo	Name	Marks

Instances: When information is inserted or deleted from the database then the database gets changed. The collection of information at particular moment is called instances. For example - following is an instance of student database

RollNo	Name	Marks
10	AAA	43
20	BBB	67

Types of Schema: The database has several schema based on the levels of abstraction.

(1) Physical Schema: The physical schema is a database design described at the physical level of abstraction.

(2) Logical Schema: The logical schema is a database design at the logical level of abstraction.

(3) Subschema:A database may have several views at the view level which are called subschemas.

Data Models

AU: Dec.-14, May-19, Marks 13

Definition: It is a collection of conceptual tools for describing data, relationships among data, semantics (meaning) of data and constraints.

• Data model is a structure below the database.

• Data model provides a way to describe the design of database at physical, logical and view level.

• There are various data models used in database systems and these are as follows -

(1) Relational model:

• Relation model consists of collection of tables which stores data and also guilatxo represents the relationship among the data.

• Table is also known as relation.

• The table contains one or more columns and each column has unique name.

• Each table contains record of particular type, and each record type defines a fixed number of fields or attributes.

• For example - Following figure shows the relational model by showing the relationship between Student and Result database. For example - Student Ram lives in city Chennai and his marks are 78. Thus the relationship between these two databases is maintained by the ***SeatNo.*** Column

SeatNo	Name	City
101	Ram	Chennai
102	Shyam	Pune

SeatNo	Marks
101	78
102	95

Advantages:

(i) Structural Independence: Structural independence is an ability that allows us to make changes in one database structure without affecting other. The relational levsiz model have structural independence. Hence making required changes in thedatabase is convenient in relational database model.

(ii)Conceptual Simplicity: The relational model allows the designer to simply focus on logical design and not on physical design. Hence relational models are conceptually simple to understand.

(iii) Query Capability: Using simple query language (such as SQL) user can get egile information from the database or designer can manipulate the database structure.

(iv) Easy design,maintenance and usage: The relational models can be designed logically hence they are easy to maintain and use.

Disadvantages:

(i) Relational model requires powerful hardware and large data storage devices.

(ii) May lead to slower processing time.

(iii) Poorly designed systems lead to poor implementation of database systems.

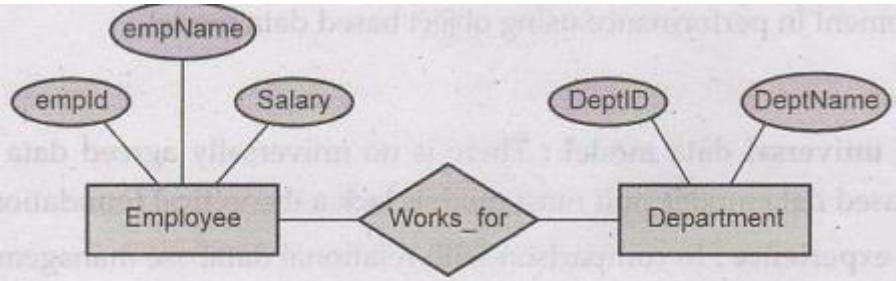
1) Entity relationship model:

• As the name suggests the entity relationship model uses collection of basic objects called entities and relationships.

• The entity is a thing or object in the real world.

• The entity relationship model is widely used in database design.

• For example - Following is a representation of Entity Relationship modelin which the relationship works_for is between entities Employee and Department.



Advantages:

i) Simple: It is simple to draw ER diagram when we know entities and relationships.

ii) Easy to understand: The design of ER diagram is very logical and hence they are easy to design and understand.

iii) Effective: It is effective communication tool.

iv) Integrated: The ER model can be easily integrated with Relational model.

v) Easy conversion: ER model can be converted easily into other type of models.

Disadvantages:

i) Loss of information: While drawing ER model some information can be hidden or lost.

ii) Limited relationships: The ER model can represent limited relationships as compared to other models.

iii) No Representation for data manipulation: It is not possible to represent data manipulation in ER model.

iv) No industry standard: There is no industry standard for notations of ER diagram.

(3) Object Based Data Model:

• The object oriented languages like C++, Java, C# are becoming the

dominant in software development.

• This led to object based data model.

• To The object based data model combines object oriented features with relationaldata model.

Advantages:

i) Enriched modelling: The object based data model has capability of modelling the real world objects.

ii) Reusability: There are certain features of object oriented design such as inheritance, polymorphism which help in reusability.

iii) Support for schema evolution: There is a tight coupling between data and b applications, hence there is strong support for schema evolution.

iv)Improved performance: Using object based data model there can be significant improvement in performance using object based data model.

Disadvantages:

i) Lack of universal data model: There is no universally agreed data model for an object based data model, and most models lack a theoretical foundation.

ii) Lack of experience: In comparison with relational database management the use of object based data model is limited. This model is more dependent on the skilled egi programmer.

iii) Complex: More functionalities present in object based data model make the design complex.

(4) Semi-structured data model:

• The semi-structured data model permits the specification of data where individual data items of same type may have different sets of attributes.

• The Extensible Markup Language (XML) is widely used to represent semi- structured data model.

Advantages

i) Data is not constrained by fixed schema.

ii) It is flexible.

iii) It is portable.

Disadvantages

i) Queries are less efficient than other types of data model.

Review Questions

1. Write short note on: Data model and its types. ***AU: Dec.-14, Marks 8***

2 Explain three different groups of data models with suitable examples. ***AU: May-19, Marks 13***

Database System Architecture work

AU: May-12,13,14,16,17, Dec.- 08,15,17,19, Marks 16

• The typical structure of typical DBMS is based on relational data model as shown in Fig 1.5.1.

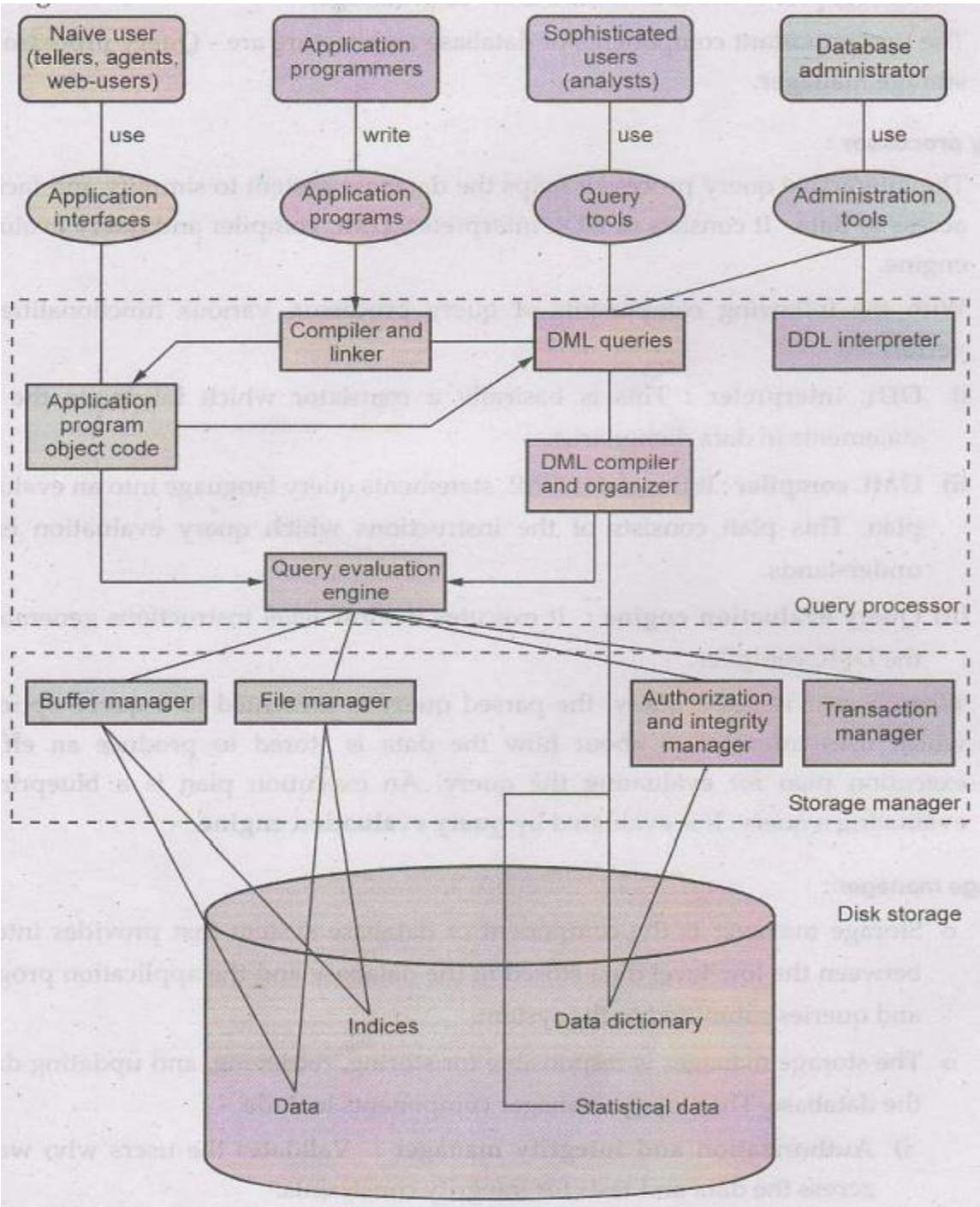


Fig. 1.5.1 : Architecture of database

• Consider the top part of Fig. 1.5.1. It shows application interfaces used by naïve users, application programs created by application programmers, query tools used by sophisticated users and administration tools used by database administrator.

• The lowest part of the architecture is for disk storage.

• The two important components of database architecture are - Query processor and storage manager.

Query processor:

• The interactive query processor helps the database system to simplify and facilitate access to data. It consists of DDL interpreter, DML compiler and query evaluation engine.

• With the following components of query processor, various functionalities are performed -

i) DDL interpreter: This is basically a translator which interprets the DDL statements in data dictionaries.

ii) DML compiler: It translates DML statements query language into an evaluation plan. This plan consists of the instructions which query evaluation engine understands.

iii) Query evaluation engine: It executes the low-level instructions generated by the DML compiler.

• When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query. An execution plan is a blueprint for evaluating a query. It is evaluated by query evaluation engine.

Storage manager:

• Storage manager is the component of database system that provides interface between the low level data stored in the database and the application programs and queries submitted to the system.

• The storage manager is responsible for storing, retrieving, and updating datain the database. The storage manager components include -

i) Authorization and integrity manager: Validates the users who want to access the data and tests for integrity constraints.

ii) Transaction manager: Ensures that the database remains in consistent despite of system failures and concurrent transaction execution proceeds without conflicting.

iii) File manager: Manages allocation of space on disk storage and representation of the information on disk.

iv) Buffer manager: Manages the fetching of data from disk storage into main memory. The buffer manager also decides what data to cache in main memory. Buffer manager is a crucial part of database system.

• Storage manager implements several data structures such as -

i) Data files: Used for storing database itself.

ii) Data dictionary: Used for storing metadata, particularly schema of database.

iii) Indices: Indices are used to provide fast access to data items present in the database

Review Questions

1. Explain the overall architecture of database system in detail. ***AU: May-14,17, Dec.-17, Marks 8, May-16, Marks 16***

2. With the help of a neat block diagram explain basic architecture of a database management system. ***AU May-12, May-13, Marks 16, Dec-15, Marks 8***

3. Explain component modules of a DBMS and their interactions with the architecture ***AU: Dec.-08, Marks 10***

4. Sketch the typical component modules of DBMS. Indicate and explain interactions between those modules of the system. ***AU: Dec.-19, Marks 7***

Introduction to Relational Databases

• Relation database is a collection of tables having unique names.

• For example - Consider the example of Student table in which the information about the student is stored.

RollNo	Name	Phone
001	AAA	1111111111
002	BBB	2222222222
003	CCC	3333333333

Fig. 1.8.1 : Student table

The above table consists of three column headers RollNo, Name and Phone. Each row of the table indicates the information of each student by means of his Roll Number, Name and Phone number.

Similarly consider another table named Course as follows –

CourseID	CourseName	Credits
101	Mechanical	4
102	Computer Science	6
103	Electrical	5
104	Civil	3

Fig. 1.8.2 : Course table

Clearly, in above table the columns are CourseID, CourseName and Credits. The CourseID 101 is associated with the course named Mechanical and associated with the course of mechanical there are 4 credit points. Thus the relation is represented by the table in the relation model. Similarly we can establish the relationship among the two tables by defining the third table. For example - Consider the table Admission as

RollNo	CourseID
001	102
002	104
003	101

Fig. 1.8.3 : Admission

From this third table we can easily find out that the course to which the RollNo 001 is admitted is computer Science.

Relational Model

There are some commonly used terms in Relational Model and those are -

Table or relation: In relational model, table is a collection of data items arranged in rows and columns. The table cannot have duplicate data or rows. Below is an example of student table

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333
004	DDD	67	4444444444

Tuple or record or row: The single entry in the table is called tuple. The tuple represents a set of related data. In above Student table there are four tuples. One of the tuple can be represented as

001	AAA	88	1111111111

Attribute or columns: It is a part of table that contains several records. Each record can be broken down into several small parts of data known as attributes. For example the above table consists of four attributes such as RollNo,Name,Marks and Phone.

Relation schema: A relation schema describes the structure of the relation, with the name of the relation (i.e. name of table), its attributes and their names and type.

Relation Instance: It refers to specific instance of relation i.e. containing a specific set of rows. For example - the following is a relation instance - which contains the records with marks above 80.

RollNo	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333

Domain: For each attribute of relation, there is a set of permitted values called domain. For example - in above table, the domain of attribute Marks is set of all possible permitted marks of the students. Similarly the domain of Name attribute is all possible names of students.

That means Domain of Marks attribute is (88,83,98)

Atomic: The domain is atomic if elements of the domain are considered to be indivisible units. For example in above Student table, the attribute Phone is non-atomic.

NULL attribute: A null is a special symbol, independent of data type, which means either unknown or inapplicable. It does not mean zero or blank. For example - Consider a salary table that contains NULL

Emp#	Job Name	Salary	Commission
E10	Sales	12500	32090
E11	Null	25000	8000
E12	Sales	44000	0
E13	Sales	44000	Null

Degree: It is nothing but total number of columns present in the relational database. In given Student table –

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333

The degree is 4.

Cardinality: It is total number of tuples present i in the relational database. In above given table the cardinality is 3

Example 1.9.1 *Find out following for given Staff table*

i) No of Columns

ii) No of tuples

iii) Different attributes

iv) Degree

v) Cardinality

StaffID	Name	Sex	Designation	Salary	DOJ
S001	John	M	Manager	50000	1 Oct. 2012
S002	Ram	M	Executive	20000	20 Jan. 2015
S003	Meena	F	Supervisor	40000	12 Aug. 2011

Solution:

i) No of Columns = 6

ii) No of Tuples = 3

iii) Different attributes are StaffID, Name, Sex, Designation, Salary, DOJ

iv) Degree Total number of columns = 6

v) Cardinality =Total number of rows = 3

Domain and Key Constraint

Domain Constraint

A domain is defined as the set of all unique values permitted for an attribute. For example, a domain of date is the set of all possible valid dates, a domain of Integer is all possible whole numbers, and a domain of day-of-week is Monday, Tuesday Sunday.

This in effect is defining rules for a particular attribute. If it is determined that an attribute is a date then it should be implemented in the database to prevent invalid dates being entered.

Domain constraints are user defined data type and we can define them like this: Domain constraint = Data type + Constraints

The constraints can be specified using NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY/CHECK/DEFAULT.

For example-

Create domain id_value integer

constraint id_test

check(value > 100); ← cheking if stud_id value is greater than 100

create table student (

stu_id id_value PRIMARY KEY,

stu_name CHAR(30),

stu_age integer

);

Key Constraint

• A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.

• For example - Consider the students relation and the constraint that no two students have the same student id. This IC is an example of a key constraint.

• The definition of key constraints contain two parts -

• Two distinct tuples in a legal instance (an instance that satisfies all Integrity Constraints including the key constraint) cannot have identical values in all the fields of a key.

• No subset of the set of fields in a key is a unique identifier for a tuple.

• The first part of the definition means that, in any legal instance, the values inthe key fields uniquely identify a tuple in the instance. When specifying a key constraint, the DBA or user must be sure that this constraint will not prevent them from storing a 'correct' set of tuples. For example, several students may have the same name, although each student has a unique student id. If the name field is declared to be a key, the DBMS will not allow the Students relation to contain two tuples describing different students with the same name.

• The second part of the definition means, for example, that the set of fields (RollNo, Name} is not a key for Students, because this set properly contains the key {RollNo}. The set {RollNo, Name} is an example of a superkey, which is a set of fields that contains a key.

• The key constraint can be specified using SQL as follows -

• In SQL, we can declare that a subset of the columns of a table constitute a key by using the UNIQUE constraint.

• At most one of these candidate keys can be declared to be a primary key, using the PRIMARY KEY constraint. For example -

CREATE TABLE Student (RollNo integer,

Name CHAR(20),

age integer,

UNIQUE (Name, age),

CONSTRAINT StudentKey PRIMARY KEY(RollNo))

This definition says that RollNo is a Primary key and Combination of Name and age is also a key.

Integrity Constraints

Database integrity means correctness or accuracy of data in the database. A database may have number of integrity constraints. For example -

(i) The Employee ID and Department ID must consists of two digits.

(ii) Every Employee ID must start with letter.

The integrity constraints are classified based on the concept of primary key and foreign key. Let us discuss the classification of constraints based on primary key and foreign key as follows-

Entity Integrity Rule

This rule states that "In the relations, the value of attribute of primary key can not be null".

The NULL represents a value for an attribute that is currently unknown or is not applicable for this tuple. The Nulls are always to deal with incomplete or exceptional data.

The primary key value helps in uniquely identifying every row in the table. Thus if the users of the database want to retrieve any row from the table or perform any action on that table, they must know the value of the key for that row. Hence it is necessary that the primary key should not have the NULL value.

Referential Integrity Rule

• Referential integrity refers to the accuracy and consistency of data within a relationship.

• In relationships, data is linked between two or more tables. This is achieved by having the foreign key (in the associated table) reference a primary key value (in the primary or parent -

table). Because of this, we need to ensure that data on both sides of the relationship remain intact.

• The referential integrity rule states that "whenever a foreign key value is used it must reference a valid, existing primary key in the parent table".

• **Example:**Consider the situation where you have two tables Employees and Managers. The Employees table has a foreign key attribute entitled Managed By, which points to the record for each employee's manager in the Managers table.

Referential integrity enforces the following three rules:

i) You cannot add a record to the Employees table unless the Managed By attribute points to a valid record in the Managers table. Referential integrity prevents the insertion of incorrect details into a table. Any operation that doesn't satisfy referential integrity rule fails.

ii) If the primary key for a record in the Managers table changes, all corresponding records in the Employees table are modified.

iii) If a record in the Managers table is deleted, all corresponding records in the Employees table are deleted.

Advantages of Referential Integrity

Referential integrity offers following advantages:

i) Prevents the entry of duplicate data.

ii) Prevents one table from pointing to a nonexistent field in another table.

ii) Guaranteed consistency between "partnered" tables.

iii) Prevents the deletion of a record that contains a value referred to by a foreign key in olds another table.

iv) Prevents the addition of a record to a table that contains a foreign key unless there is etail a primary key in the linked table.

Review Question

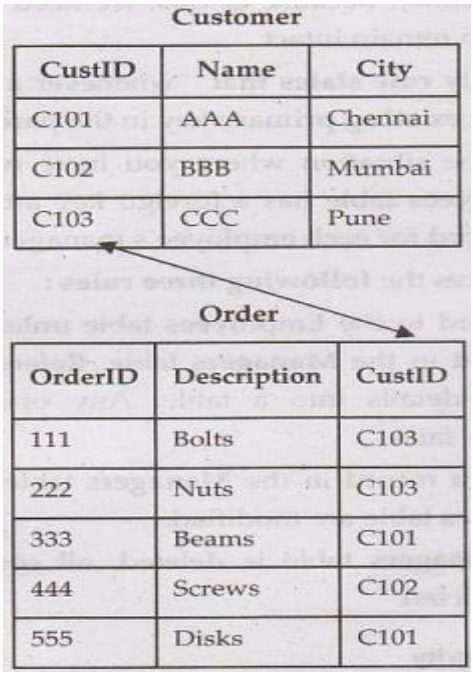
1. Discuss the entity Integrity and referential integrity constraints. Why are they important? Explain them with suitable examples.

Database integrity

• The foreign key is a key in one table that refers to the primary key of another table.

• The foreign key is basically used to link two tables. For example

Consider Customer table as follows –



Customer

CustID	Name	City
C101	AAA	Chennai
C102	BBB	Mumbai
C103	CCC	Pune

Order

OrderID	Description	CustID
111	Bolts	C103
222	Nuts	C103
333	Beams	C101
444	Screws	C102
555	Disks	C101

• Note that the "CustID" column in the "Order" table points to the "CustID" column in the "Customer" table.

• The "CustID" column in the "Customer" table is the PRIMARY KEY in the"Customer" table.

• The "CustID" column in the "Order" table is a FOREIGN KEY in the "Order" table.

• The table containing the foreign key is called the child table, and the table containing the primary key is called the referenced or parent table.

• The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

• The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

Relational Algebra

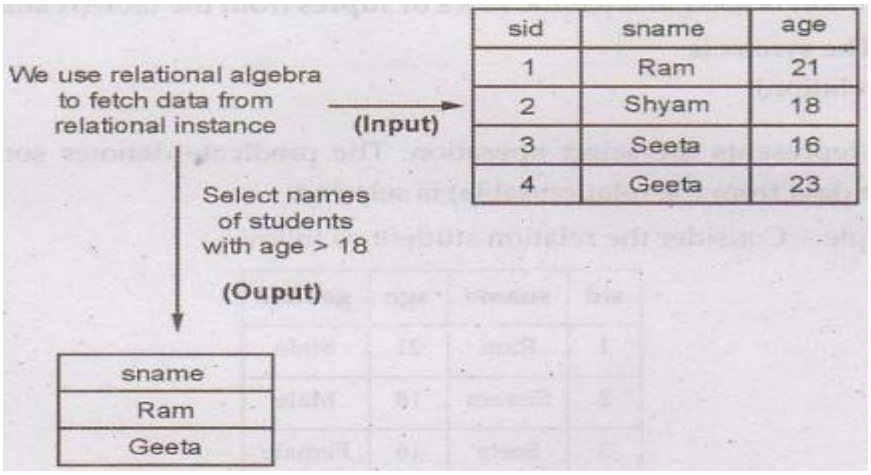
AU: May-03,04,05,14,15,16,17,18, Dec.-02,06,07,08,11,15,16,17, Marks 16

There are two formal query languages associated with relational model and those are relational algebra and relational calculus.

• **Definition:** Relational algebra is a procedural query language which is used to access database tables to read data in different ways.

• The queries present in the relational algebra are denoted using operators.

• Every operator in relational algebra accepts relational instances (tables) as input and returns relational instance as output. For example-

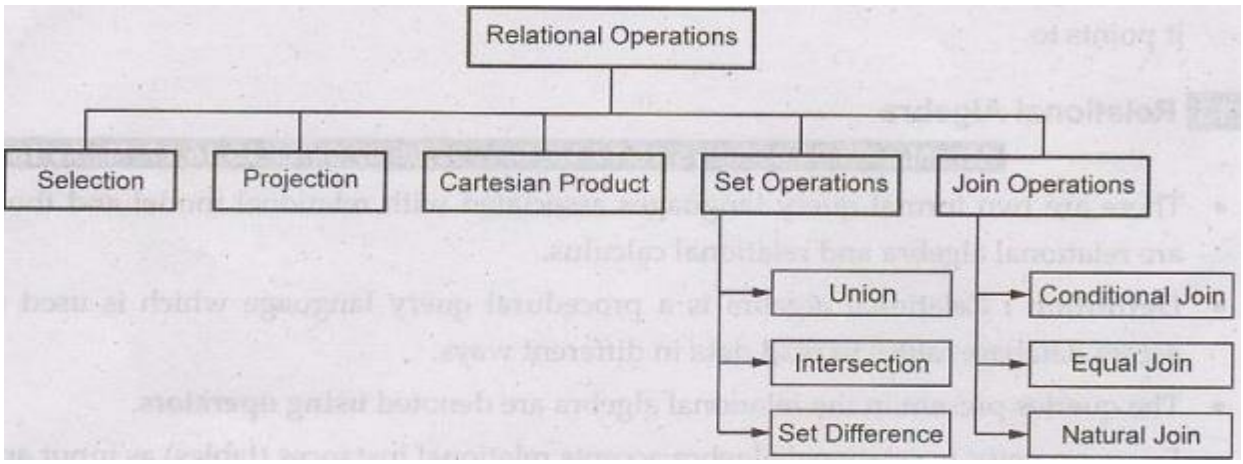


• Each relational algebra is procedural. That means each relational query describes a step-by-step procedure for computing the desired answer, based on the order in which operators are applied in the query.

• A sequence of relational algebra operations forms a relational algebra expression, whose result will also be a relation that represents the result of a database query. The By composing the operators in relational expressions the complex relation can be defined.

Relational Operations

Various types of relational operations are as follows-



(1) Selection:

• This operation is used to fetch the rows or tuples from the table(relation).

• Syntax: The syntax is

$\sigma_{predicate}$(relation)

Where σrepresents the select operation. The predicate denotes some logic using which the data from the relation (table) is selected.

• For example - Consider the relation student as follows-

sid	sname	age	gender
1	Ram	21	Male
2	Shyam	18	Male
3	Seeta	16	Female
4	Geeta	23	Female

Fig. 1.13.1 : Student Table

Query: Fetch students with age more than 18

We can write it in relational algebra as

$\Sigma_{age>>18}$ (student)

The output will be-

sname
Ram
Geeta

We can also specify conditions using and, or operators.

$\sigma_{age>18 \text{ and } gender = 'Male'}$ (Student)

sname
Ram

(2)Projection :

• Project operation is used to project only a certain set of attributes of a relation. That means if you want to see only the names all of the students in the Student table, then you can use Project operation.

• Thus to display particular column from the relation, the projection operator is used.

• It will only project or show the columns or attributes asked for, and will also vait remove duplicate data from the columns.

• **Syntax:**

ΠC1, C2... (r)

where C1, C2 etc. are attribute names(column names).

• For example - Consider the Student table given in Fig. 1.13.2.

Query: Display the name and age all the students

This can be written in relational algebra as

$\Pi_{sname,\ age}$ (Student)

Above statement will show us only the Name and Age columns for all the rows of data in Student table.

sname	age
Ram	21
Shyam	18
Seeta	16
Geeta	23

Fig. 1.13.2

(3) Cartesian product:

• This is used to combine data from two different relations(tables) into one and fetch data from the combined relation.

• Syntax: A x B

• **For example:** Suppose there are two tables named Student and Reserve as follows

Student

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

Query: Find the names of all the students who have reserved isbn = 005. To satisfy this query we need to extract data from two table. Hence the cartesian product operator is used as

($\sigma_{Student.sid = Reserve.sid \wedge Reserve.Isbn = 005}$ (Student × Reserve)

As an output we will get

sid	sname	age	sid	isbn	day
1	Ram	21	1	005	07-07-18
2	Shyam	18	2	005	03-03-18

Note:that although the Sid columns is same, it is repeated.

(4) Set operations: Various set operations are - union, intersection and set-difference. Let us understand each of these operations with the help of examples.

(i) Union:

• This operation is used to fetch data from two relations(tables) or temporary relation(result of another operation).

• For this operation to work, the relations(tables) specified should have same number of attributes(columns) and same attribute domain. Also the duplicate tuples are automatically eliminated from the result.

• **Syntax: AUB**

• Where A and B are relations.

• **For example:** If there are two tables student and books as follows-

Student

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Book

isbn	bname	Author
005	DBMS	XYZ
006	OS	PQR
007	DAA	ABC

• **Query:** We want to display both the student name and book names from both the tables then

Π_{sname} (Student) U Π_{bname} (Book)

(ii) Intersection:

• This operation is used to fetch data from both tables which is common in both the tables.

• **Syntax: A ∩ B**

where A and B are relations.

• Example - Consider two tables - Student and Worker

Student

Name	Branch
AAA	ComputerSci
BBB	Mechanical
CCC	Civil
DDD	Electrical

Worker

Name	Salary
XXX	3000
AAA	2000
YYY	1500
DDD	2500

• **Query:** If we want to find out the names of the students who are working in a company then 300

Π_{name} (Student) ∩ Π_{name} (Worker)

Name
AAA
DDD

(iii) Set-Difference: The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Syntax: A - B

For Example: Consider two relations Full_Time_Employee and Part_Time_Employee, if we want to find out all the employee working for Fulltime, then the set difference operator is used -

$\Pi_{EmpName}$(Full Time_Employee) – $\Pi_{EmpName}$(Part_Time_Employee)

(5) Join: The join operation is used to combine information from two or more relations. Formally join can be defined as a cross-product followed by selections and projections, joins arise much more frequently in practice than plain cross-products. The join operator is used as

A) Inner Join

There are three types of joins used in relational algebra

i) Conditional join: This is an operation in which information from two tables is combined using some condition and this condition is specified along with the join operator.

$A \bowtie_c B = \sigma_c (A \times B)$

Thus ⋈ is defined to be a cross-product followed by a selection. Note that the condition c can refer to attributes of both A and B. The condition C can be specified using <,<,>,< or = operators.

For example consider two table student and reserve as follows-

Student

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

If we want the names of students with sid(Student) = sid (Reserve) and isbn = 005,then we can write it using Cartesian product as -

(σ((Student.sid = Reserve.sid) ∩(Reserve.(isbn) =005)) (Student × Reserve))

Here there are two conditions as

i) (Student.sid =Reserve.sid) and ii) (Reserve.isbn = 005) which are joined by∩operator.

Now we can use $\bowtie_c$ instead of above statement and write it as -

(Student $\bowtie_{(\text{Student.sid - Reserve.sid}) \wedge (\text{Reserve.(Isbn) - 005})}$ Reserve))

The result will be-

sid	sname	age	isbn	day
1	Ram	21	005	07-07-18
2	Shyam	18	005	03-03-18

ii) Equijoin: This is a kind of join in which there is equality condition between two attributes(columns) of relations(tables). For example - If there are two table Book and Reserve table and we want to find the book which is reserved by the student having isbn 005 and name of the book is 'DBMS', then :

Book

isbn	bname	Author
005	DBMS	XYZ
006	OS	PQR
007	DAA	ABC

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

(Obname='DBMS' (Book (Book.isbn = Reserve.isbn) Reserve)

Then we get

isbn	bname	Author	sid	day
005	DBMS	XYZ	1	07-07-18
005	DBMS	XYZ	2	03-03-18

iii) Natural Join: When there are common columns and we have to equate these common columns then we use natural join. The symbol for natural join is simply ⋈ without any condition. For example, consider two tables-

Book

isbn	bname	Author
005	DBMS	XYZ
006	OS	PQR
007	DAA	ABC

Reserve

sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

Now if we want to list the books that are reserved, then that means we want to match Books.isbn with Reserve.isbn. Hence it will be simply Books Reserve

B) Outer Join

There are three types of outer joins - Left outer join, Right outer join and Full outer join.

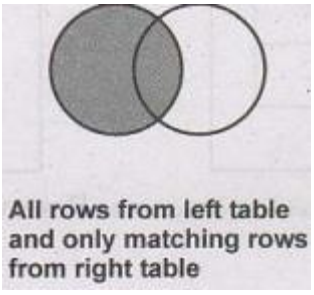
(1) Left Outer join

• This is a type of join in which all the records from left table are returned and the matched records from the right table gets returned.

• The result is NULL from the right side, if there is no match.

• The symbol used for left outer join is ⟕

• This can be graphically represented as follows



• For example - Consider two tables Student and Course as follows –

Student

RollNo	Name
1	AAA
2	BBB
3	CCC

Course

RollNo	CourseName
1	Computer
3	Electrical
5	Civil
6	Mechanical

The result of Left outer join will be

RollNo	Name	CourseName
1	AAA	Computer
2	BBB	NULL
3	CCC	Electrical

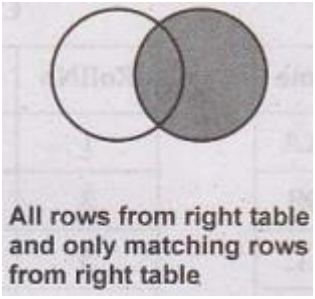
(2) Right Outer join

• This is a type of join in which all the records from right table are returned and the matched records from the left table gets returned.

• The result is NULL from the left side, if there is no match.

• The symbol used for right outer join is ⟖

• This can be graphically represented as follows



• **For example** - Consider two tables Student and Course as follows-

Student

RollNo	Name
1	AAA
2	BBB
3	CCC

Course

RollNo	CourseName
1	Computer
3	Electrical
5	Civil
6	Mechanical

The result of Right outer join will be

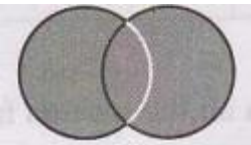
RollNo	CourseName	Name
1	Computer	AAA
3	Electrical	CCC
5	Civil	NULL
6	Mechanical	NULL

(3) Full Outer join

• In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

• It is denoted by

• Graphically it can be represented as



• **For example** - Consider two tables Student and Course as follows –

Student

RollNo	Name
1	AAA
2	BBB
3	CCC

Course

RollNo	CourseName
1	Computer
3	Electrical
5	Civil
6	Mechanical

The result of Right outer join will be

RollNo	Name	CourseName
1	AAA	Computer
2	BBB	NULL
3	CCC	Electrical
5	NULL	Civil
6	NULL	Mechanical

The symbols used are –

A LEFT OUTER JOIN B ON x=y	$A \bowtie_{x=y} B$
A RIGHT OUTER JOIN B ON x=y	$A \bowtie_{x=y} B$
A FULL OUTER JOIN B ON x=y	$A \bowtie_{x=y} B$

(6) Rename operation: This operation is used to rename the output relation for any query operation which returns result like Select, Project etc. Or to simply rename a relation(table). The operator ρ(rho) is used for renaming.

Syntax: ρ (RelationNew, RelationOld)

For example: If you want to create a relation Student_names with sid and sname from Student, it can be done using rename operator as:

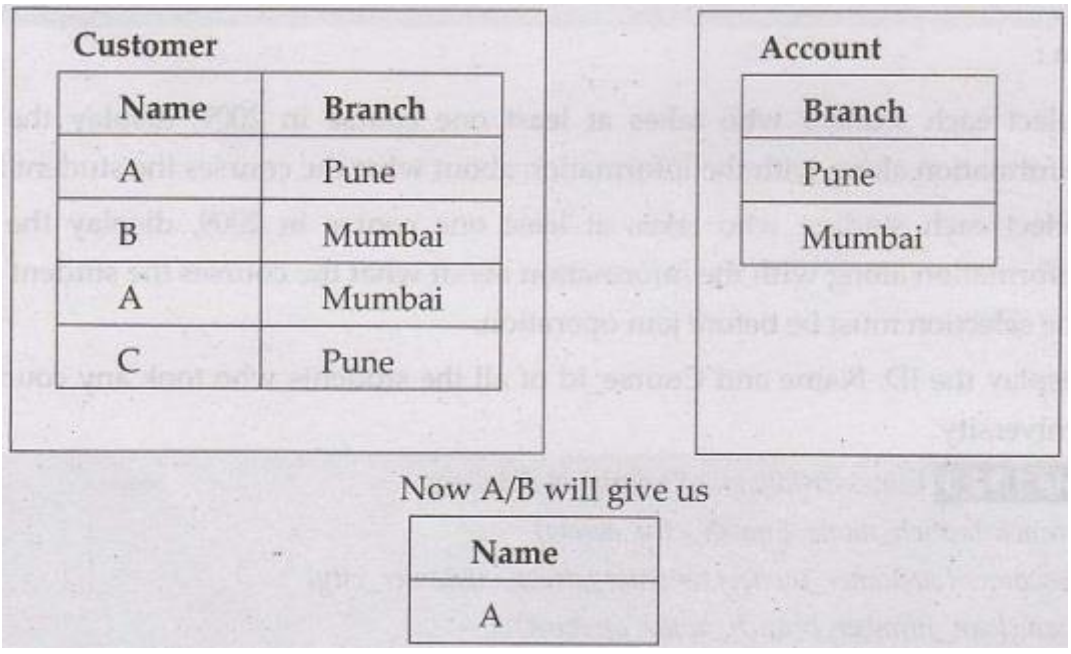
$\rho(\text{Student\_names}, (\Pi_{\text{sid.sname}}(\text{Student}))$

(7) Divide operation: The division operator is used when we have to evaluate queries which contain the keyword ALL.

It is denoted by A/B where A and B are instances of relation.

Formal Definition of Division Operation: The operation A/B is define as the set of all x values (in the form of unary tuples) such that for every y value in (a tuple of) B, there is a tuple <x,y> in A.

For example - Find all the customers having accounts in all the branches. For that consider two tables - Customer and Account as



Customer

Name	Branch
A	Pune
B	Mumbai
A	Mumbai
C	Pune

Account

Branch
Pune
Mumbai

Now A/B will give us

Name
A

Here We check all the branches from Account table against all the names from Customer table. We can then find that only customer A has all the accounts in all the branches.

Review Questions

1. Explain select, project, cartesian product and join operations in relational algebra with an example lebeidi enne roberts **AU: May-18, Marks 13, Dec.-16, Marks 6**

2. List operations of relational algebra and purpose of each with example. **AU: May-17, Marks 5**

3.Differentiate between foreign key constraints and referential integrity constraints with suitable example. **AU: Dec.-17, Marks 6**

4. Explain various operations in relational algebra with examples

AU: May-03, Marks 10, Dec-07, Marks 8, Dec.- 08, Marks 10, May-14, Marks 16

5. Explain all join operations in relational algebra. **AU: May-05, Marks 8**

6. Briefly explain relational algebra.

AU: May-04, Marks 8

7. What is rename operation in relational algebra ? Illustrate your answer with example.

AU: Dec.-02, Marks 2

SQL Fundamentals

AU: Dec.-14,15,17,19, May-15,16,17,18, Marks 15

• Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS.

• Various parts of SQL are -

• **Data Definition Language(DDL):** It consists of a set of commands for defining relation schema, deleting relations, and modifying relation schemas.

• **Data Manipulation Language(DML):** It consists of set of SQL commands for inserting tuples into relational schema, deleting tuples from or modifying tuples in databases.

• **Integrity:** The SQL DDL includes commands for specifying integrity constraints. These constraints must be satisfied by the databases.

• **View definition:** The SQL DDL contains the commands for defining views for database.

• **Transaction control:** The SQL also includes the set of commands that indicate beginning and ending of the transactions.

• **Embedded SQL and Dynamic SQL:** There is a facility of including SQL commands in the programming languages like C,C++, COBOL or Java.

• **Authorization:** The SQL DDL includes the commands for specifying access rights to relations and views.

Data Abstraction

The Basic data types used in SQL are -

(1) char(n): For representing the fixed length character string this data type is used. For instance to represent name, designation, coursename, we use this data type. Instead of char we can also use character. The n is specified by the user.

(2) varchar(n): The varchar means character varying. That means - for denoting the variable length character strings this data type is used. The n is user specified maximum character length.

(3) int: For representing the numeric values without precision, the int data type is used.

(4) numeric: For representing, a fixed point number with user-specified precision this data type is used. The number consists of m digits plus sign k digits are to the right of precision. For instance the numeric(3,2) allows 333.11 but it does not allow 3333.11

(5) smallint: It is used to store small integer value. It allows machine dependent subset of integer type.

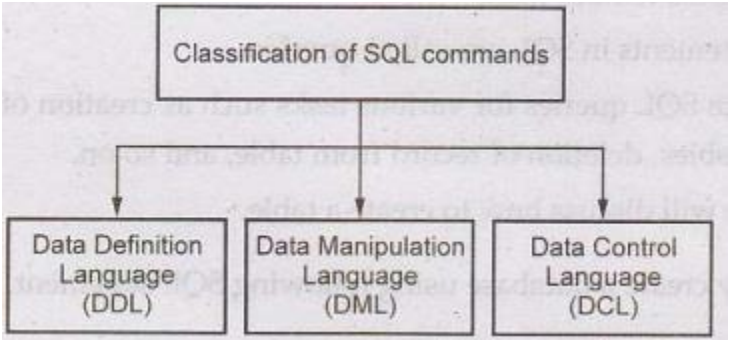
(6) real: It allows the floating point, double precision numbers.

(7) float(n): For representing the floating point number with precision of at least n digits this data type is used.

Basic Schema Definition

In this section, we will discuss various SQL commands for creating the schema definition.

There are three categories of SQL commands.



Data Definition Language

Sr.No.	Command	Purpose
1.	CREATE	This command is used to create database, tables, views or any other database objects.
2.	ALTER	It modifies the existing tables.
3.	DROP	This command deletes complete table, view or any other database object.

Data Manipulation Language

Sr. No.	Command	Purpose
1.	SELECT	This command is used to retrieve either all or desired records from one or more tables.
2.	INSERT	For inserting the records in the table, this command is used.
3.	UPDATE	For updating one or more fields of the table, this command is used.
4.	DELETE	This command is used for deleting the desired record.

Data Control Language

Sr. No.	Command	Purpose
1.	GRANT	This command is used to give access rights or privileges to the database.
2.	INVOKE	The revoke command removes user access rights or privileges to the database objects.

1. Creation

• A database can be considered as a container for tables and a table is a grid with rows and columns to hold data.

• Individual statements in SQL are called queries.

• We can execute SQL queries for various tasks such as creation of tables, insertion of data into the tables, deletion of record from table, and so on.

In this section we will discuss how to create a table.

Step 1: We normally create a database using following SQL statement..

Syntax

CREATE DATABASE database_name;

Example

CREATE DATABASE Person _DB

Step 2: The table can be created inside the database as follows -

CREATE TABLE table name (

Col1_name datatype,

col2 _name datatype,

……

coln_name datatype

);

Example

CREATE TABLE person_details{

AdharNo int,

FirstName VARCHAR(20),

MiddleName VARCHAR(20),

LastName VARCHAR(20),

Address VARCHAR(30),

City VARCHAR(10)

}

The blank table will be created with following structure

Person_details

AdharNo	FirstName	MiddleName	LastName	Address	City

2. Insertion

We can insert data into the table using INSERT statement.

Syntax

INSERT INTO table_name (col_1, col_2,...,col_n)

VALUES ($value_1$,value,...., $value_n$)

Example

INSERT INTO person_details (AdharNo, FirstName, MiddleName, LastName, Address, City)

VALUES (111, 'AAA','BBB','CCC','M.G. Road', 'Pune')

The above query will result into –

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune

3. Select

• The Select statement is used to fetch the data from the database table.

• The result returns the data in the form of table. These result tables are called resultsets.

• We can use the keyword DISTINCT. It is an optional keyword indicating that the answer should not contain duplicates. Normally if we write the SQL without DISTINCT operator then it does not eliminate the duplicates.

Syntax

SELECT col1, col2, ...,coln FROM table_name;

Example

SELECT AdharNo, FirstName, Address, City from person_details

The result of above query will be

AdharNo	FirstName	City
111	AAA	Pune

• If we want to select all the records present in the table we make use of * character.

Syntax

SELECT FROM table_name;

Example

*SELECT * FROM person_details;*

The above query will result into

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune

4. Where Clause

The WHERE command is used to specify some condition. Based on this condition the data present in the table can be displayed or can be updated or deleted.

Syntax

SELECT col1,col2, ...,coln

FROM table_name

WHERE condition;

Example

Consider following table-

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

If we execute the following query

SELECT AdharNo

FROM person_details

WHERE city='Pune';

The result will be

AdharNo	City
111	Pune
222	Pune

If we want records of all those person who live in city Pune then we can write the query using WHERE clause as

*SELECT **

FROM person_details

WHERE city='Pune';

The result of above query will be

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune

5. Update

• For modifying the existing record of a table, update query is used.

Syntax

UPDATE table name

SET col1-value1, col2-value2,...

WHERE condition;

Example

Consider following table

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani Chowk	Delhi
444	JJJ	KKK	LLL	Viman Nagar	Mumbai

Person_details table

If we execute following query

UPDATE rerson_details

SET city 'Chennai'

WHERE AdharNo=333

The result will be

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani Chowk	Delhi
444	JJJ	KKK	LLL	Viman Nagar	Mumbai

6. Deletion

We can delete one or more records based on some condition. The syntax is as follows -

Syntax

DELETE FROM table_name WHERE condition;

Example

DELETE FROM person_details

WHERE AdharNo=333

The result will be –

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
444	JJJ	KKK	LLL	Viman nagar	Mumbai

We can delete all the records from table. But in this deletion, all the records get deleted without deleting table. For that purpose the SQL statement will be

DELETE FROM person_details;

7. Logical Operators

• Using WHERE clause we can use the operators such as AND, OR and NOT.

• AND operator displays the records if all the conditions that are separated using AND operator are true.

• OR operator displays the records if any one of the condition separated using OR operator is true.

• NOT operator displays a record if the condition is NOT TRUE.

Consider following table

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

Syntax of AND

SELECT col1, col2, ...

FROM table_name

WHERE condition1 AND condition2 AND condition3...;

Example of AND

If we execute following query-

SELECT AdharNo, FirstName, City

FROM person_details

WHERE AdharNo=222 AND City= 'Pune';

The result will be –

AdharNo	FirstName	City
222	DDD	Pune

Syntax of OR

SELECT col1, col2, ...

FROM table_name

WHERE condition1 OR condition2 OR condition3 ...;

Example of OR

SELECT AdharNo, FirstName, City

FROM person_details

WHERE City='Pune' OR City='Mumbai';

The result will be –

AdharNo	FirstName	City
111	AAA	Pune
222	DDD	Pune
444	JJJ	Mumbai

Syntax of NOT

SELECT col1, col2, ...

FROM table_name

WHERE NOT condition;

Example of NOT

SELECT AdharNo, FirstName, City

FROM person_details

WHERE NOT City='Pune';

The result will be

AdharNo	FirstName	City
333	GGG	Delhi
444	JJJ	Mumbai

8. Order By Clause

• Many times we need the records in the table to be in sorted order.

• If the records are arranged in increasing order of some column then it is called ascending order.

• If the records are arranged in decreasing order of some column then it is called descending order.

• For getting the sorted records in the table we use ORDER BY command.

• The ORDER BY keyword sorts the records in ascending order by default.

Syntax

SELECT col1, col2,...,coln

FROM table_name

ORDER BY col1,col2.... ASC | DESC

Here ASC is for ascending order display and DESC is for descending order display.

Example

Consider following table

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

*SELECT **

FROM person_details

ORDER BY AdharNo DESC;

The above query will result in

AdharNo	FirstName	MiddleName	LastName	Address	City
444	JJJ	KKK	LLL	Viman nagar	Mumbai
333	GGG	HHH	III	Chandani chowk	Delhi
222	DDD	EEE	FFF	Shivaji nagar	Pune
111	AAA	BBB	CCC	M.G. road	Pune

9. Alteration

There are SQL command for alteration of table. That means we can add new column or delete some column from the table using these alteration commands.z

Syntax for Adding columns

ALTER TABLE table_name

ADD column_name datatype;

Example

Consider following table

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

If we execute following command

ALTER TABLE Customers

ADD Email varchar(30);

Then the result will be as follows –

AdharNo	FirstName	MiddleName	LastName	Address	City	Email
111	AAA	BBB	CCC	M.G. road	Pune	NULL
222	DDD	EEE	FFF	Shivaji nagar	Pune	NULL
333	GGG	HHH	III	Chandani chowk	Delhi	NULL
444	JJJ	KKK	LLL	Viman nagar	Mumbai	NULL

Syntax for Deleting columns

ALTER TABLE table_name

DROP COLUMN column name;

Example

Consider following table

AdharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

If we execute following command

ALTER TABLE Customers

DROP COLUMN Address;

Then the result will be as follows –

AdharNo	FirstName	MiddleName	LastName	City
111	AAA	BBB	CCC	Pune
222	DDD	EEE	FFF	Pune
333	GGG	HHH	III	Delhi
444	JJJ	KKK	LLL	Mumbai

10. Defining Constraints

• We can specify rules for data in a table.

• When the table is created at that time we can define the constraints.

• The constraint can be column level i.e. we can impose constraint on the column and table level i.e we can impose constraint on the entire table.

There are various types of constraints that can be defined are as follows -

1) Primary key: The primary key constraint is defined to uniquely identify the records from the table.

The primary key must contain unique values. Hence database designer should choose primary key very carefully.

For example

Consider that we have to create a person_details table with AdharNo, FirstName, MiddleName, LastName, Address and City.

Now making AdharNo as a primary key is helpful here as using this field it becomes easy to identify the records correctly.

The result will be

CREATE TABLE person_details (

AdharNo int,

FirstName VARCHAR(20),

MiddleName VARCHAR(20),

LastName VARCHAR(20),

Address VARCHAR(30),

City VARCHAR(10),

PRIMARY KEY(AdharNo)

);

We can create a composite key as a primary key using CONSTRAINT keyword. For example

CREATE TABLE person_details (

AdharNo int NOT NULL,

FirstName VARCHAR(20),

MiddleName VARCHAR(20),

LastName VARCHAR(20) NOT NULL,

Address VARCHAR(30),

City VARCHAR(10),

CONSTRAINT PK_person_details PRIMARY KEY(AdharNo, LastName)

);

(2) Foreign Key

• Foreign key is used to link two tables.

• Foreign key for one table is actually a primary key of another table.

• The table containing foreign key is called child table and the table containing candidate primary key is called parent key.

• Consider

Employee Table

EmpID	LastName	FirstName	Age
1	Khanna	Rajesh	30
2	Joshi	Sharman	23
3	Kapoor	Tushar	20

Dept Table:

DeptID	DeptName	EmpID
1	Accounts	3
2	Production	3
3	Sales	2
4	Purchase	1

• Notice that the "EmpID" column in the "Dept" table points to the "EmpID" column in the "Employee" table.

• The "EmpID" column in the "Employee" table is the PRIMARY KEY in the "Employee" table.

• The "EmpID" column in the "Dept" table is a FOREIGN KEY in the "Dept" table.

• The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

• The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

• The purpose of the foreign key constraint is to enforce referential integrity but there are also performance benefits to be had by including them in your database design.

The table Dept can be created as follows with foreign key constraint.

CREATE TABLE DEPT (

DeptID int

DeptName VARCHAR(20),

EmpID int,

PRIMARY KEY(DeptID),

FOREIGN KEY (EmpID)

REFERENCES EMPLOYEE(EmpID)

);

(3) Unique

Unique constraint is used to prevent same values in a column. In the EMPLOYEE table, for example, you might want to prevent two or more employees from having an identical designation. Then in that case we must use unique constraint.

We can set the constraint as unique at the time of creation of table, or if the table is already created and we want to add the unique constraint then we can use ALTER command.

For example -

CREATE TABLE EMPLOYEE(

EmpID INT NOT NULL,

Name VARCHAR (20) NOT NULL,

Designation VARCHAR(20) NOT NULL UNIQUE,

Salary DECIMAL (12, 2),

PRIMARY KEY (EmpID)

);

If table is already created then also we can add the unique constraint as follows -

ALTER TABLE EMPLOYEE

MODIFY Designation VARCHAR(20) NOT NULL UNIQUE;

(4) NOT NULL

• By default the column can have NULL values.

• NULL means unknown values.

• We can set the column values as non NULL by using the constraint NOT NULL.

• For example

CREATE TABLE EMPLOYEE(

EmpID INT NOT NULL,

Name VARCHAR (20) NOT NULL,

Designation VARCHAR(20) NOT NULL,

Salary DECIMAL (12, 2) NOT NULL,

PRIMARY KEY (EmpID)

);

(5) CHECK

The CHECK constraint is used to limit the value range that can be placed in a column.

For example

CREATE TABLE parts (

Part_no int PRIMARY KEY,

Description VARCHAR(40),

Price DECIMAL(10, 2) NOT NULL CHECK(cost > 0)

);

(6) IN operator

The IN operator is just similar to OR operator.

It allows to specify multiple values in WHERE clause.

Syntax

SELECT col1,col2,....

FROM table_name

WHERE column-name IN (value1, value2,...);

Example

Consider following table

Employee

empID	empName	Salary	DeptID
1	AAA	1000	D101
2	BBB	2000	D102
3	CCC	3000	D103
4	DDD	4000	D104
5	EEE	5000	D105

SELECT FROM Employee

WHERE empID IN (1, 3);

The result will be

empID	empName	Salary	DeptID
1	AAA	1000	D101
2	BBB	2000	D102
3	CCC	3000	D103

Basic Structure of SQL Queries

The basic form of SQL queries is

SELECT-FROM-WHERE. The syntax is as follows:

SELECT[DISTINCT] target-list

FROMRelation-list

WHEREQualification

• **SELECT:** This is one of the fundamental query command of SQL. It is similar to the projection operation of relational algebra. It selects the attributes based on the condition described by WHERE clause.

• **FROM:** This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given, this clause corresponds to Cartesian product.

• **WHERE:** This clause defines predicate or conditions, which must match in order to qualify the attributes to be projected.

• **Relation-list:** A list of relation names(tables)

• **Target-list:** A list of attributes of relations from relation list(tables)

• **Qualification:**Comparisons of attributes with values or with other attributes combined using AND, OR and NOT.

• **DISTINCT** is an optional keyword indicating that the answer should not contain duplicates. Normally if we write the SQL without DISTINCT operator then it does not eliminate the duplicates.

Example

SELECT sname

FROM Student

WHERE age>18

• The above query will return names of all the students from student table where age of each student is greater than 18

Queries on Multiple Relations

Many times it is required to access multiple relations (tables) to operate on some information. For example consider two tables as Student and Reserve.

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

sid	isbn	day
1	005	07-07-18
2	007	03-03-18
3	009	

Query: Find the names of students who have reserved the books with book isbn

Select Student.sname,Reserve.isbn

From Student, Reserve

Where Student.sid=Reserve.sid

Use of SQL Join

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

Example: Consider two tables for using the joins in SQL. Note that cid is common column in following tables.

Student

sid	cid	sname
1	101	Ram
2	101	Shyam
3	102	Seeta
4	NULL	Geeta

Reserve

cid	cname
101	Pune
102	Mumbai
103	Chennai

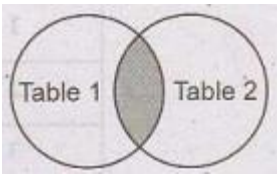
1) Inner Join:

• The most important and frequently used of the joins is the INNER JOIN. They are also known as an EQUIJOIN.

• The INNER JOIN creates a new result table by combining column values of two alqutul no tables (Table1 and Table2) based upon the join-predicate.

• The query compares each row of tablel with each row of Table2 to find all pairs of rows which satisfy the join-predicate.

• When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. It can be represented as:



• **Syntax:** The basic syntax of the INNER JOIN is as follows.

SELECT Table1.column1, Table2.column2...

FROM Table1

INNER JOIN Table2

ON Table1.common_field = Table2.common_field;

• **Example:** For above given two tables namely Student and City, we can apply inner join. It will return the record that are matching in both tables using the common column cid. The query will be

*SELECT **

FROM Student Inner Join City on Student.cid=City.cid

The result will be

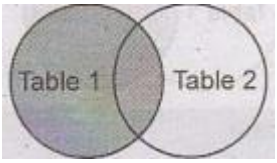
sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai

2) Left Join:

• The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in the right table; the join will still return a row in the result, but with NULL in each column from the right table.

• This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.

• It can be represented as –



• **Syntax:** The basic syntax of a LEFT JOIN is as follows.

SELECT

SELECT Table1.column1, Table2.column2...

FROM Table1

LEFT JOIN Table2

ON Table1.common field Table2.common field;

• **Example:** For above given two tables namely Student and City, we can apply Left join. It will Return all records from the left table, and the matched records from the right table using the common column cid. The query will be

*SELECT **

FROM Student Left Join City on Student.cid=City.cid

The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL

3) Right Join:

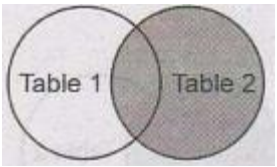
• The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table.

• This means that if the ON clause matches 0 (zero) records in the left table; the join will still return a row in the result, but with NULL in each column from the left table.

• This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

• It can be represented as follows:

• **Syntax:** The basic syntax of a RIGHT JOIN is as follow-



SELECT Table1.column1, Table2.column2...

FROM Table1

RIGHT JOIN Table2

ON Table1.common_field = Table2.common_field;

• **Example:** For above given two tables namely Student and City, we can apply Rightjoin. It will return all records from the right table, and the matched records from the left table using the common column cid. The query will be,

*SELECT **

FROM Student Right Join City on Student.cid=City.cid

The result will be –

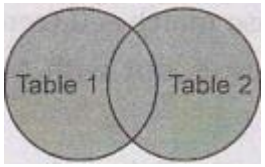
sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
NULL	NULL	NULL	103	Chennai

4) Full Join:

• The SQL FULL JOIN combines the results of both left and right outer joins.

• The joined table will contain all records from both the tables and fill in NULLS for missing matches on either side.

• It can be represented as



• **Syntax:** The basic syntax of a FULL JOIN is as follows

SELECT Table1.column1, Table2.column2...

FROM Table1 FULL JOIN Table2 ON Table1.common_field = Table2.common_field;

The result will be -

Example: For above given two tables namely Student and City, we can apply Full join. It will return returns rows when there is a match in one of the tables using the common column cid. The query will be -

*SELECT **

FROM Student Full Join City on Student.cid=City.cid

The result will be –

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL
NULL	NULL	NULL	103	Chennai

Database Languages

There are two types of languages supported by database systems. These are

(1) DDL

• Data Definition Language (DDL) is a specialized language used to specify a database schema by a set of definitions.

• It is a language which is used for creating and modifying the structures of tables, views, indexes and so on.

• DDL is also used to specify additional properties of data.

• Some of the common commands used in DDL are - CREATE, ALTER, DROP.) The main use of CREATE command is to build a new table. Using ALTER command, the users can add up some additional column and drop existing columns. Using DROP command, the user can delete table or view.

(2) DML

• DML stands for Data Manipulation Language.

• This language enables users to access or manipulate data as organized by appropriate data model.

• The types of access are-

• **Retrieval** of information stored in the database.

• **Insertion** of new information into the database.

• **Deletion** of information from the database.

• **Modification** of information stored in database.

• There are two types of DML -

• **Procedural DML -** Require a user to specify what data are needed and how to get those data.

• **Declarative DML** - Require a user to specify what data are needed without of au aw specifying how to get those data.

• **Query** is a statement used for requesting the retrieval of information. This retrieval of information using some specific language is called query language.

Review Question

1. Briefly explain about views of data. **AU: May-16, Marks 16**

Keys

AU: May-06, 07, 12, Dec.-06, Marks 4

Keys are used to specify the tuples distinctly in the given relation.

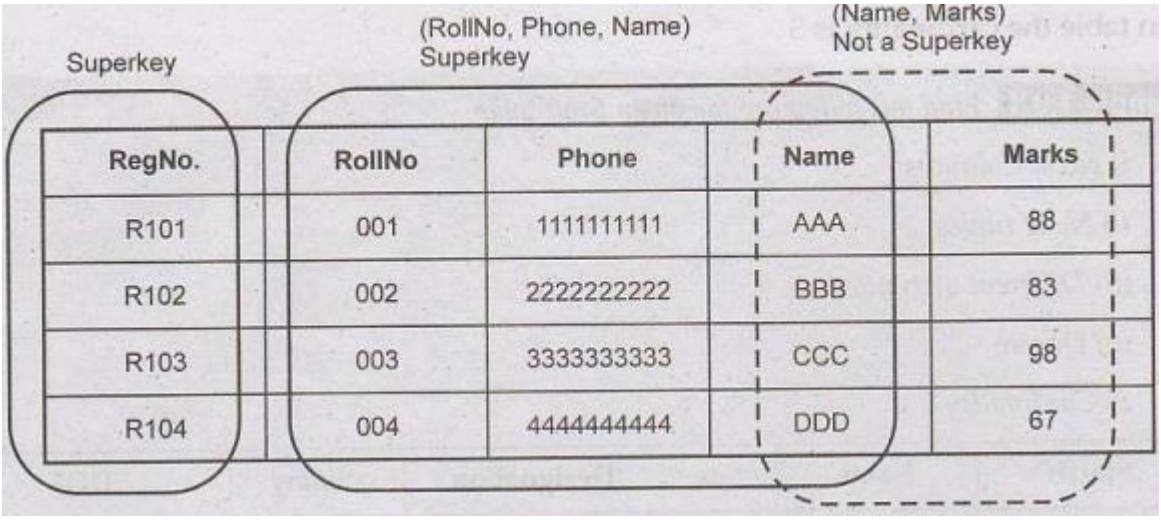
Various types of keys used in relational model are - Superkey, Candidate Keys, primary keys, foreign keys. Let us discuss them with suitable example

1) Super Key(SK): It is a set of one or more attributes within a table that can uniquely identify each record within a table. For example - Consider the Student table as follows-

Reg No.	Roll No	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

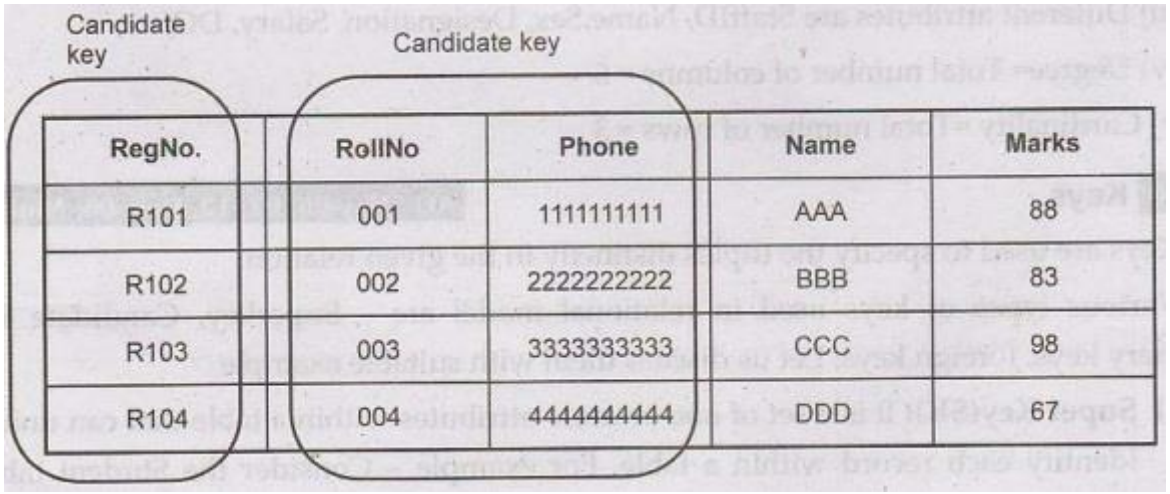
Fig. 1.10.1 : Student

The superkey can be represented as follows-



Clearly using the (RegNo) and (RollNo, Phone,Name) we can identify the records uniquely but (Name, Marks) of two students can be same, hence this combination not necessarily help in identifying the record uniquely.

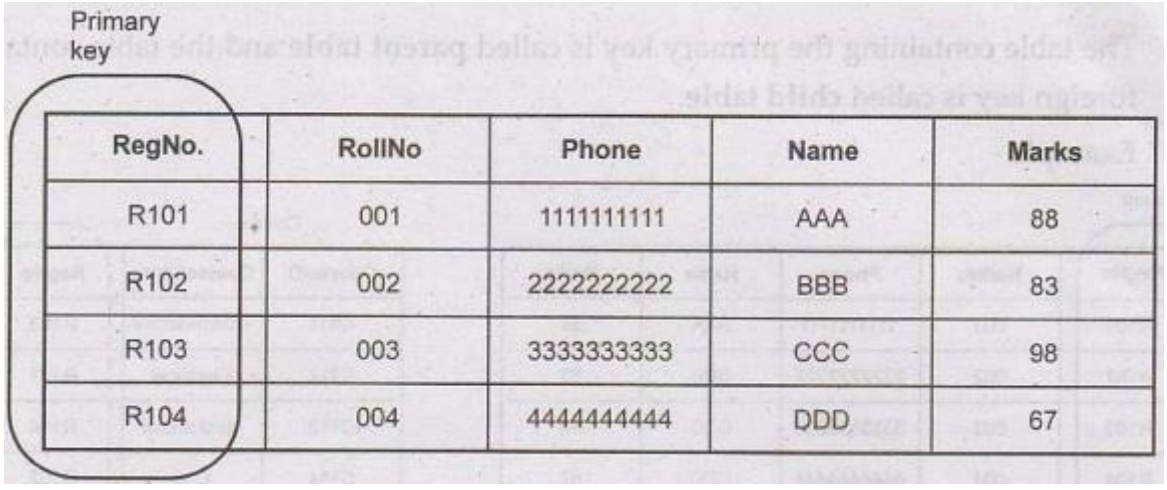
2) Candidate Key(CK): The candidate key is a subset of superset. In other words candidate key is a single attribute or least or minimal combination of attributes that uniquely identify each record in the table. For example - in above given Student table, the candidate key is RegNo, (RollNo,Phone). The candidate key can be



RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Thus every candidate key is a superkey but every superkey is not a candidate key.

3) Primary Key(PK): The primary key is a candidate key chosen by the database designer to identify the tuple in the relation uniquely. For example - Consider the following representation of primary key in the student table



RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Other than the above mentioned primary key, various possible primary keys can be (RollNo), (RollNo, Name), (RollNo, Phone)

The relation among super key, candidate key and primary can be denoted by

Candidate Key= Super Key- Primary Key

Rules for Primary Key

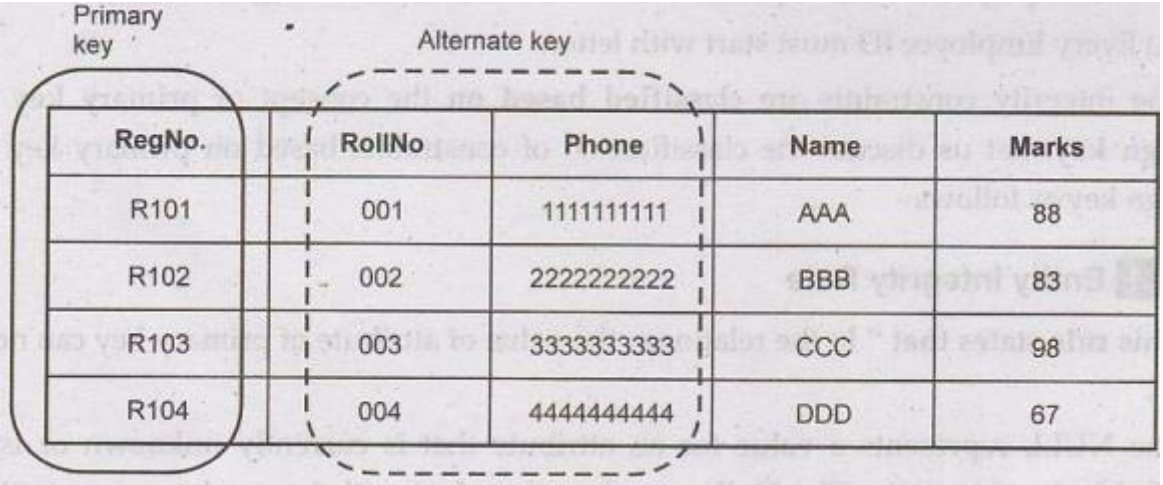
(i) The primary key may have one or more attributes.

(ii) There is only one primary key in the relation.

(iii) The value of primary key attribute can not be NULL.

(iv) The value of primary key attributes does not be NULL.

4) Alternate Key: The alternate key is a candidate key which is not chosen by the database designer to uniquely identify the tubles. For example-



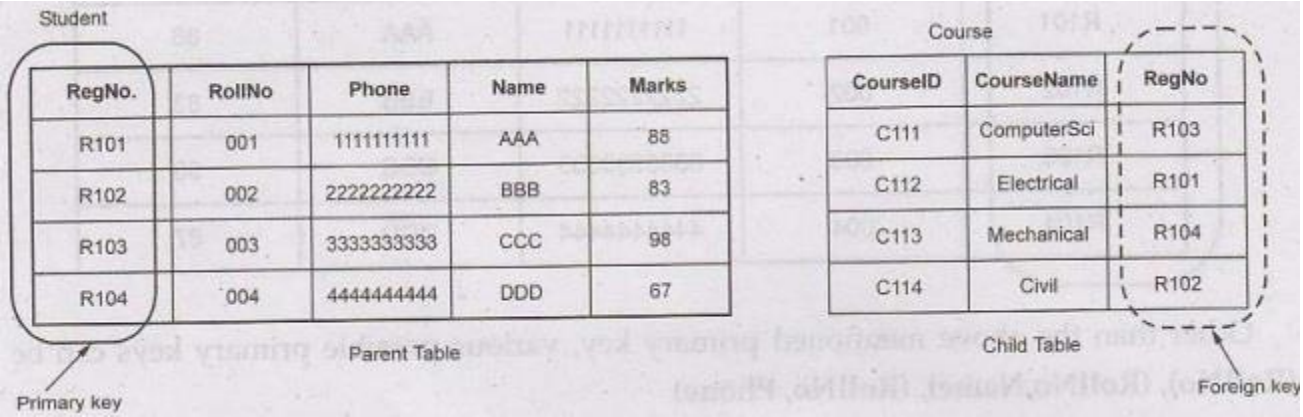
RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

5) Foreign key: Foreign key is a single attribute or collection of attributes in one table that refers to the primary key of other table.

• Thus foreign keys refers to primary key.

• The table containing the primary key is called parent table and the table containing foreign key is called child table.

• Example-



RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

CourseID	CourseName	RegNo
C111	ComputerSci	R103
C112	Electrical	R101
C113	Mechanical	R104
C114	Civil	R102

From above example, we can see that two tables are linked. For instance we could easily find out that the 'Student CCC has opted for ComputerSci course

Review Question

*1. Explain distinction among the terms primary key, candidate key, foreign key and super key with suitable example****. AU: May-06, 07, 12, Dec.-06, Marks 4 AU: Dec.-05, Marks 10***