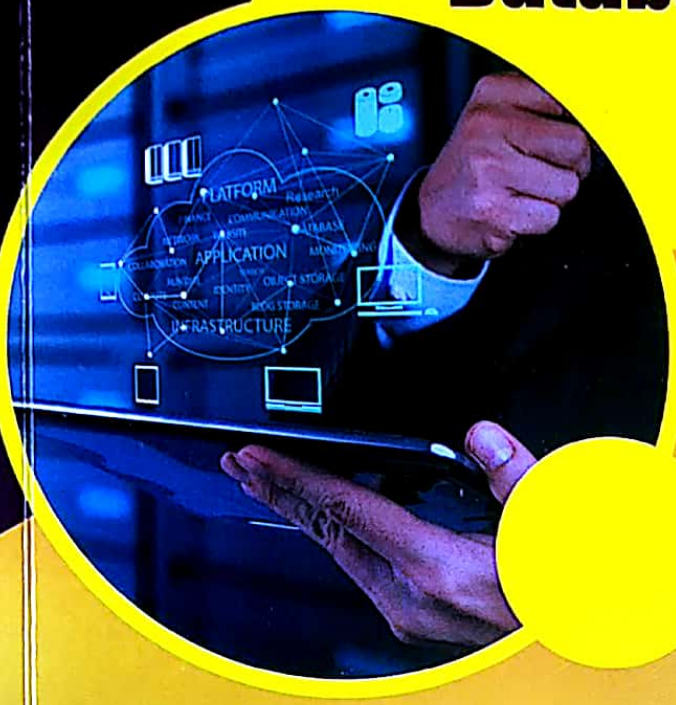
Database Management Systems



Sub Code : 3130703

- AS PER NEW SYLLABUS OF GUJARAT TECHNOLOGICAL UNIVERSITY
- [illegible]III (CE/IT)
- [illegible]IFIED APPROACH
- ORAL QUESTIONS AND ANSWERS
- CHAPTERWISE SOLVED GTU QUESTIONS JUNE 2005 to WINTER 2018

MODEL PAPER
AS PER NEW QUESTION PAPER PATTERN

TECHNICAL PUBLICATIONS
An Up-Thrust for Knowledge

A. A. Puntambekar

TABLE OF CONTENTS

1 Database System Architecture

Syllabus

Data Abstraction, Data Independence, Data Definition Language (DDL), Data Manipulation Language (DML).

Contents

1.1 Basic Database Management System Concepts

GTU : Winter-14, 15, Summer-17, Marks 5

1.1.1 Introduction to Database Management System (DBMS)

- **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** being 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 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.

1.1.2 Purpose of Database System

- 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 need for addition of new data, 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 difficult 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 specific 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 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 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.
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. *Define DBMS and list out purpose of DBMS.* **GTU : Winter-15, Mark 1**
2. *Write the advantages of Database System* **GTU : Summer-17, Mark 1**
3. *How DBMS is better than file management system.* **GTU : Winter-15, Marks 4**
4. *Explain advantages and disadvantages of conventional file-based system over database management system.* **GTU : Winter-14, Marks 5**

1.2 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.
- The database administrators decide how to store data in physical level.
- 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.
- For example – A Clerk at the reservation system, can see only part of the database and can access the required information of the passenger.

Fig. 1.2.1 shows the relationship among the three levels of abstraction.

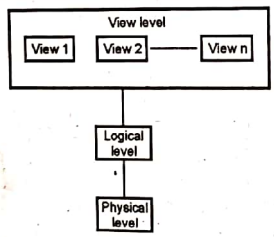


Fig. 1.2.1 : Levels of data abstraction

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 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.

1.3 Database System Architecture

GTU : Summer-18, Winter-14,18, Marks 7

- The typical structure of typical DBMS is based on relational data model as shown in Fig. 1.3.1.
- Consider the top part of Fig. 1.3.1. 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.

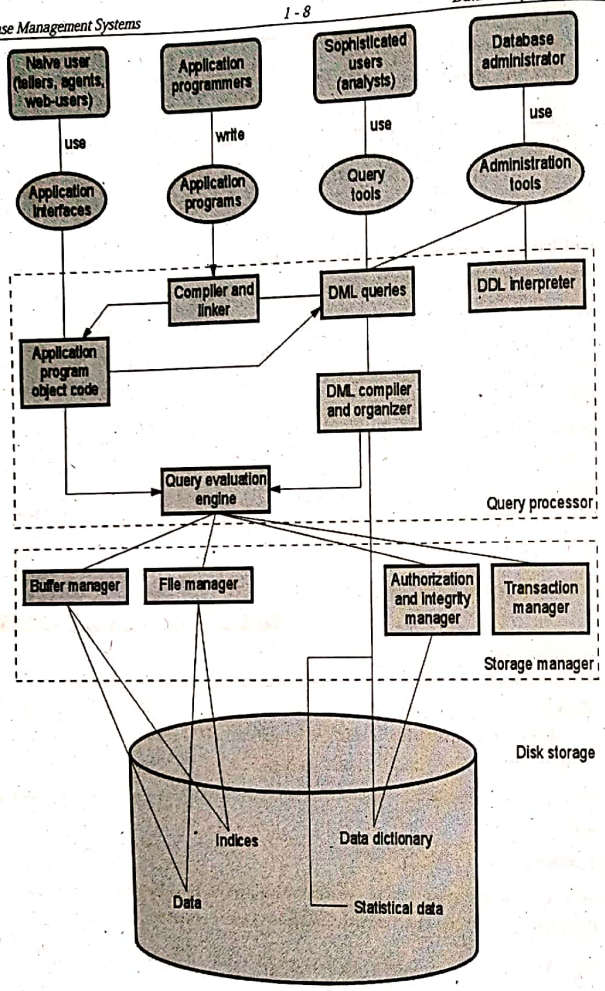


Fig. 1.3.1 Architecture of database

- 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 data** in 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 state** 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 Question

1. *Explain three levels architecture of DBMS.* **GTU : Summer-18, Winter-14,18. Marks 7**

1.4 Data Independence

GTU : Winter-13, Marks 7

- **Definition :** Data independence is an ability by which one can change the data at one level without affecting the data at another level. Here level can be physical, conceptual or external.
- Data independence is one of the important characteristics of database management system.
- By this property, the structure of the database or the values stored in the database can be easily modified by without changing the application programs.
- There are two types of data independence.

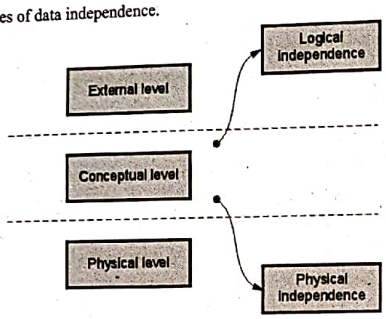


Fig. 1.4.1 Data independence

1. **Physical Independence :**
   - This is a kind of data independence which allows the modification of physical schema without requiring any change to the conceptual schema.
   - For example - if there is any change in memory size of database server then it will not affect the logical structure of any data object.
2. **Logical Independence :**
   - This is a kind of data independence which allows the modification of conceptual schema without requiring any change to the external schema.
   - For example - Any change in the table structure such as addition or deletion of some column does not affect user views.

By these data independence the time and cost acquired by changes in any one level can be reduced and abstract view of data can be provided to the user.

Review Question

1. *What is data independence ? Explain different types of data independence with suitable example*

GTU : Winter-13, Marks 7

1.5 Instances and Schema

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.

1.6 Data Definition Language (DDL)

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

Data Definition Language and Data manipulation Language

- 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.

1.7 Data Manipulation Language (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 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. *Explain the concept of DDL and DML.*

1.8 Oral Questions and Answers

Q.1 What is Database Management System ? Why do we need a DBMS ?

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

Q.2 What is the purpose of database management system ?

Ans. : The purpose of database management system is to -

- **Define the structure** for storage of information.
- Provide mechanism for **manipulation of information.**
- In addition, the database systems must ensure the **safety of information** stored.

Q.3 List any two advantages of database systems

Ans. : Following are the advantages of DBMS -

1) DBMS **removes the data redundancy** that means there is no duplication of data in database.
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.

Q.4 Define data abstraction

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

Q.5 What are three levels of data abstraction ?

Ans. : The three levels of data abstraction are -

1. Physical Level
2. Logical Level
3. View Level

Q.6 Name the categories of SQL commands

Ans. : The categories of SQL commands are -

(1) Data Definition Language (DDL)

(2) Data Manipulation Language (DML)

(3) Data Control Language (DCL)

Q.7 Define data independence.

Ans. : Data independence is an ability by which one can change the data at one level without affecting the data at another level. Here level can be physical, conceptual or external.

Q.8 What is meant by instance and Schema of the database

Ans. : • When information is inserted or deleted from the database then the database gets changed. The collection of information at particular moment is called **instances.**

- The overall design of the database is called schema.

❑❑❑

2 Data Models

Syllabus

Entity-relationship model, network model, relational and object oriented data models, integrity constraints, data manipulation operations.

Contents

2.1 Concept of Data Model

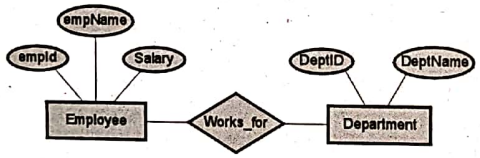
- **Definition :** Data Model is a collection of conceptual tools for describing data, relationships among data, semantics (meaning) of data and constraints.
- Data model is a **structure below the database.**
- Data model provides a way to describe the design of database at physical, logical and view level.
- There are various data models used in database systems and these are as follows -
  1) Entity Relationship Model 2) Network Model
  3) Relational Model 4) Object Oriented Data Model

Review Question

1. *What is data model ? Enlist different types of data model.*

2.2 Entity Relationship Model

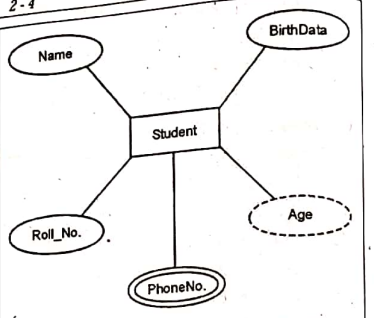
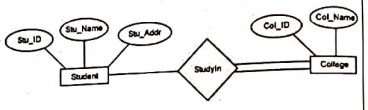
- As the name suggests the entity relationship model uses collection of basic objects called **entities** and **relationships.**
- The entity is a thing or object in the real world.
- The entity relationship model is widely used in database design.
- For example - Following is a representation of Entity Relationship model in which the relationship **works_for** is between entities **Employee** and **Department.**



Notations used in Entity Relationship Model

Component	Symbol	Example
Entity : Any real-world object can be represented as an entity about which data can be stored in a database. All		Student

the real world objects like a book, an organization, a product, a car, a person are the examples of an entity.		
Relationship : Rhombus is used to setup relationships between two or more entities.		Teachers — Teaches — Students
Attribute : Each entity has a set of properties. These properties of each entity are termed as attributes. For example, a car entity would be described by attributes such as price, registration number, model number, color etc	Attribute Key Attribute	Registration Number, Model Number, Price, Car, Color
Derived attribute : Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth. To represent a derived attribute, another dotted ellipse is created inside the main ellipse.	Derived Attribute	Name, BirthData, Student, Roll_No., Age

Multivalued attribute : An attribute that can hold multiple values is known as multivalued attribute. We represent it with double ellipses in an E-R Diagram. E.g. A person can have more than one phone numbers so the phone number attribute is multivalued.		
Total participation : Each entity is involved in the relationship. Total participation is represented by double lines.		

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.

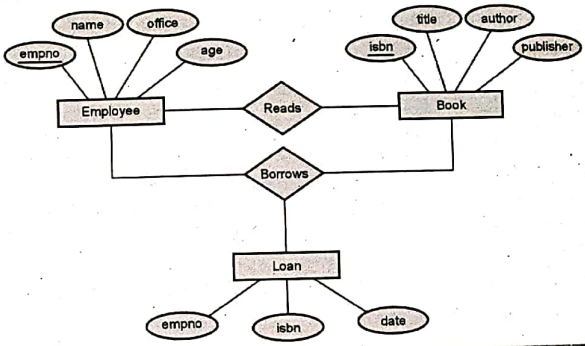
iii) **No representation for data manipulation :** It is not possible to represent data manipulation in ER model.

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

Example 2.2.1 *Consider the relation schema as given below. Design and draw an ER diagram that capture the information of this schema.*

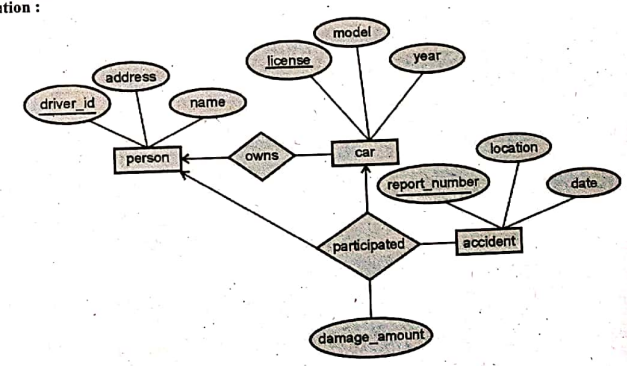
Employee(empno,name,office,age)
Books(isbn,title,authors,publisher)
Loan(empno,isbn,date)

Solution :



Example 2.2.2 *Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for particular period of time and has an associated due date and date when the payment was received.*

Solution :



Review Question

1. *Explain E-R model with suitable example*

2.3 Network Model

- The network model is a model in which data is organized in nodes and links. It resembles the graph data structure.
- In this model a record can be derived from more than one parent records.
- Entities are represented as a connected network with each other.
- The network model allows 1:1(one:one), 1:M(one:many) and M:M(many:many) relationships among entities.
- The network model can be represented using data-structure diagram. In this diagram –

  1) **Box** represents record types

  2) **Lines** represents the links
- **For example** – Consider following E-R Model for which the network data structure model can be prepared as follows -

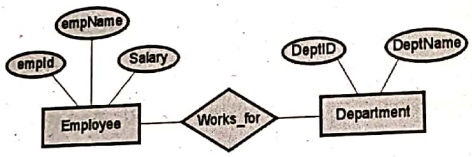


Fig. 2.3.1 E-R diagram

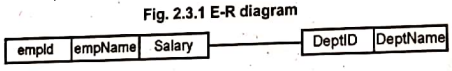
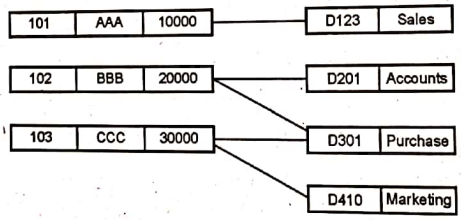


Fig. 2.3.2 Data structure model

From the above data structure model, the network model can be prepared as follows-



Advantages

1) **Simplicity :** The network model is conceptually simple and easy to design
2) **Handles more relation types :** It can handle many to many or one to many relations. One record can be associated with more than one other records.
3) **Data access :** It provides easier and flexible access to data.
4) **Data integrity :** In network, there is no isolated record. Every record exists in relationship with other record and thus the integrity or consistency of the data is maintained.
5) **Database standards :** The network model is based on standards.

Disadvantages

1) **Complex :** The overall structure becomes complex and the pointers of all the records need to be maintained.
2) **Operational anomalies :** As there are many to many association among the relationships, insertion, deletion or updation in one record may not get reflected in other record. This is called operation anomalies.

Review Question

1. *Explain network model. Give advantages and disadvantages of it.*

2.4 Relational and Object Oriented Data Models

1) Relational model :

- Relation model consists of collection of tables which stores data and also 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

Seat No	Name	City
101	Ram	Chennai
102	Shyam	Pune

Seat No	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 model have structural independence. Hence making required changes in the database 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 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.

2) Object Oriented Data Models

- The object oriented languages like C++, Java, C# are becoming the dominant in software development.
- This led to object oriented data model.
- The object oriented data model combines object oriented features with relational data 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 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 programmer.

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

2.5 Introduction to Relational Model

GTU : Winter-12, 13, Marks 4

- 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. 2.5.1 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. 2.5.2 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. 2.5.3 Admission

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

2.5.1 Commonly used Terms in 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 in the relational database. In above given table the cardinality is 3.

Example 2.5.1 *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

2.5.2 Keys

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

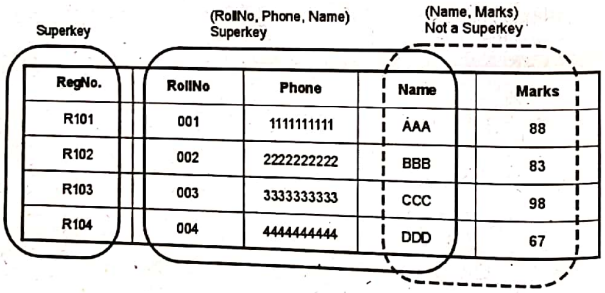
Various types of keys used in relational model are – Superkey, Candidate keys, primary keys, foreign keys. Let us discuss them with suitable example

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

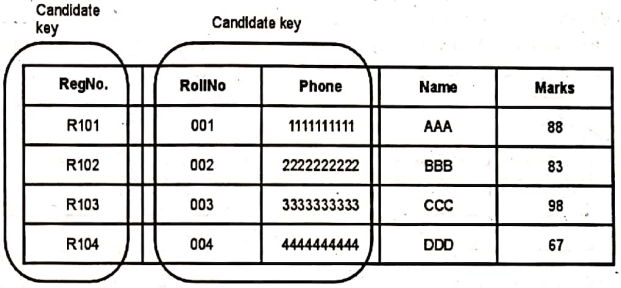
Fig. 2.5.4 Student

The superkey can be represented as follows



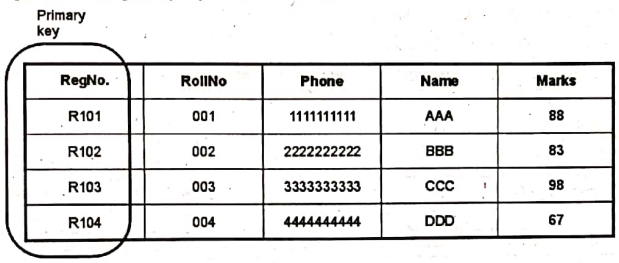
Clearly using the (RegNo) and (RollNo,Phone,Name) we can identify the records uniquely but (Name, Marks) of two students can be same, hence this combination not necessarily help in identifying the record uniquely.

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 given **Student** table, the candidate key is RegNo, (RollNo,Phone). The candidate key can be



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



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 attribute does not get changed.

4) **Alternate key :** The alternate key is a candidate key which is not chosen by the database designer to uniquely identify the tuples. For example –

Primary key — Alternate key

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 refer to primary key.
- The table containing the primary key is called **parent table** and the table containing foreign key is called **child table**.
- Example -

Student

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

Primary key — Parent Table

Course

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

Child Table — Foreign key

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 Questions

1. *What is relational model? Explain the basic terms used in it.*
2. *Define 1) Primary key 2) Foreign key 3) Unique key 4) Candidate key*

GTU : Winter-12, 13, Marks 4

2.6 Integrity Constraints

- Constraints means some rules or restrictions that are set on the database.
- Integrity constraints are rules that are to be applied on database columns to ensure the validity of data.
- For example –
  - i) The Employee ID and Department ID must consists of two digits.
  - ii) Every Employee ID must start with letter.
- Every time data is entered into that particular column, it is evaluated against the constraint and only if the result comes out to be true, then the data is inserted into the column.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- The integrity constraint thus help in maintaining the quality of database by managing the consistency among the data present in the database.
- Integrity constraints are extremely necessary for any database management system

Need for Integrity Constraint

- An Integrity Constraint (IC) is a condition specified on a database schema and restricts the data that can be stored in an instance of the database.
- Integrity constraints are used to ensure accuracy and consistency of data in a relational database.
- Data integrity is handled in a relational database through the concept of referential integrity.
- Integrity constraints ensure that changes (update deletion, insertion) made to the database by authorized users do not result in a loss of data consistency. Thus, integrity constraints guard safe against accidental damage to the database
- The integrity constraints are classified based on the concept of primary key and foreign key.
- There are four types of integrity constrains.

- Entity Integrity Constraint
- Referential Integrity Constraint
- Key Constraint
- Domain Constraint

1) Entity Integrity Constraint

- **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.
- For example –

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

NULL is not allowed!!!

2) Referential Integrity Constraint

- 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".

- For example – Consider two tables

Primary key

RegNo	Roll No	Name	Marks	Phone
R101	001	AAA	88	111111111
R102	001	BBB	83	2222222222
R103	003	CCC	98	3333333333
R104	004	DDD	67	4444444444

Student table as parent table

Relation

Foreign key

CourseID	CourseName	RegNo
c111	ComputerSci	R103
c112	Electrical	R101
c113	Mechanical	R555
c114	Civil	R102

This value is not present in parent table

Course table as child table

In above relation, the registration no. R555 is not existing still if it is present in the course table, then we say that it is not following referential integrity constraint.

3) Key Constraint

- Keys are used to identify particular record from the table. Primary key is normally used to identify the record uniquely.
- Hence the key constraint can be stated as -
  - All values of primary key must be unique.
  - The value of primary key must not be NULL.
- For example - Consider the Student table as follows. For this relation, the **Roll No** is a primary key. It is expected to find the desired record using this primary key.

Roll No	Name	Marks	Phone
001	AAA	88	111111111
001	BBB	83	2222222222
003	CCC	98	3333333333
004	DDD	67	4444444444

Duplicate values ? This is not allowed !!!

The above relation does not satisfy key constraint as the primary key is not having unique value.

4) Domain Constraint

- Domain constraint defines the domain or set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.
- For example - Consider the Student table as follows.

Roll No	Name	Marks	Phone
001	AAA	88	111111111
001	BBB	83	2222222222
003	1234	98	3333333333
004	DDD	67	4444444444

Name cannot be numeric value. It must be a string. Hence this is not allowed !!!

The above relation does not satisfy the domain constraint.

2.7 Data Manipulation Operations

GTU : Dec.-05, Summer-14, Marks 7

- The data manipulation operations are – Insert, Delete and Update(or Modify).
- The **insert** operation inserts new data in the database, delete operation deletes some data from the database and update operations makes some changes in the data present in the database. Whenever these operations are applied on the database, the **integrity constraints** specified on the relational database **must not be violated.**

Let us discuss these operations one by one

1) Insert Operation :

Consider following Student table –

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

If we perform

Insert(005, EEE,NULL,5555555555)

Then this operation will be rejected as it violates entity integrity constraint. Because the NULL value is inserted for Marks field.

If insertion violates one or more integrity constraints then the default option is to reject the insert operation.

2) Delete Operation : Consider following two tables

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

Student table as parent table

CourseID	CourseName	RegNo
c111	ComputerSci	R103
c112	Electrical	R101
c113	Mechanical	R101
c114	Civil	R102

Course table as child table

If we perform

Delete on Student table where RegNo=R101

This violates the referential integrity constraint. Because there are two tuples in Course table that may get affected if we delete the RegNo=R101(assuming that we have to delete only one student record)

The deletion operation can violate only referential integrity.

There are several options available if a deletion operation violates referential integrity.

i) One option is called **restrict** - it rejects the delete operation.

ii) Second option is **set null** or **set default**. This option either sets the value to NULL or changes the reference to valid tuples.

iii) Third option is to **cascade or propagate** the deletion by deleting tuples that reference the tuple that is being deleted.

3) Update : This operation makes changes the values of one or more attributes of tuples.

For example – Consider two tables Student table and Course table

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

Student table as parent table

CourseID	CourseName	RegNo
c111	ComputerSci	R103
c112	Electrical	R101
c113	Mechanical	R104
c114	Civil	R102

Course table as child table

If we apply

Update on RegNo of **Course** with CourseID = c112 to R555

The result of above operation is – it is rejected as it violates referential integrity. The R555 registration number is not in existence in the Parent table **Student**

Updating an attribute that is neither part of a primary key nor of a foreign key usually causes no problems.

The DBMS need only check to confirm that the new value is of the correct data type and domain.

Review Questions

1. *Explain basic concept of referential integrity.* **GTU : Dec.- 05, Marks 2**
2. *What are integrity constraints ? Explain various types of integrity constraints with suitable example.* **GTU : Summer-14, Marks 7**

2.8 Oral Questions and Answers

Q.1 What is data model ?

Ans. : 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.

Q.2 What are different types of data models ?

Ans. : Various types of data models are -

(1) Relational Data Model (2) Entity Relational Data Model

(3) Object Based Data Model (4) Semi-structured Data Model

Q.3 Explain entity relationship model.

Ans. : The ER data model specifies enterprise schema that represents the overall logical structure of a database.

The E-R model is very useful in mapping the meanings and interactions of real-world entities onto a conceptual schema.

Q.4 Give the limitations of E-R model. How do you overcome this ?

Ans. : 1) Loss of information content : Some information be lost or hidden in ER model

Limited relationship representation : ER model represents limited relationship as compared to another data models like relational model etc.

No representation of data manipulation : It is difficult to show data manipulation in ER model.

4) Popular for high level design : ER model is very popular for designing high level design.

Q.5 What is an entity ?

Ans. : " An entity is an object that exists and is distinguishable from other objects.

For example - Student named "Poonam" is an entity and can be identified by her name. Entity is represented as a box, in ER model.

Q.6 Define the term tuple.

Ans. : Tuple means a row present in the table

Q.7 Why does SQL allow duplicate tuples in a table or in a query result ?

Ans. : Data can be the same. Two people may have the same name. Since SQL is a database where you store your data and data can be duplicate.

But we can apply primary key constraints, Unique constraints or Distinct keyword to identify the record uniquely.

Q.8 Why key is essential ? Write the different types of keys

Ans. : 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.

Q.9 Define primary key. Give example.

Ans. : The primary key is a candidate key chosen by the database designer to identify the tuple in the relation uniquely.

For example - Consider a Student database as Student (RollNo,Name,Address). The primary key for this database is RollNo.The primary is underlined.

Q.10 Define foreign key. Give example

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

For example - Consider a Student database as Student (RollNo,Name,Address) and Course(CourseId, CourseName, RollNo). Here RollNo is a foreign key

Q.11 What is the difference between primary key and foreign key ?

Ans. :

Primary Key	Foreign Key
Primary key is a column or a set of columns that can be used to uniquely identify a row in a table	Foreign key is a column or a set of columns that refer to a primary key or a candidate key of another table.
A table can have a single primary key,	A table can have multiple foreign keys that can reference different tables.

Q.12 What is referential integrity ?

Ans. : 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 **ManagedBy**, which points to the record for each employee's manager in the **Managers** table.

Q.13 What is domain integrity? Give example

Ans. : Domain integrity ensures that all the data items in a column fall within a defined set of valid values. Each column in a table has a defined set of values, such as the set of all numbers for zip (five-digit), the set of all character strings for name.

Q.14 What are different types of integrity constraints used in designing relational databases

Ans. : Different types of integrity constraints are -

(1) Entity Integrity Constraint
(2) Referential Integrity Constraint
(3) Domain Integrity Constraint
(4) Key Integrity Constraint

Q.15 List the reasons why null value might be introduced into the database

Ans. : NULL is a special value provided by database in two cases –

i) When field values of some tuples are unknown(For e.g. city name is not assigned) and
ii) inapplicable(For e.g. middle name is not present).

❑❑❑

3 Relational Query Languages

Syllabus

Relational algebra, Tuple and domain relational calculus, SQL3, DDL and DML constructs, Open source and Commercial DBMS - MYSQL, ORACLE, DB2, SQL server.

Contents

3.1 Introduction to Relational Query Language

- There are **two formal languages** used for relational model and those are –
  - Relational Algebra
  - Relational Calculus
- The **relational algebra** defines a **set of operations** for the relational model, the **relational calculus** provides a **higher-level declarative language** for specifying relational queries.
- Historically the Structured Query Language **(SQL)** - The practical language used during database programming is developed later than the relation algebra and relational calculus.
- In fact, some of the concepts of SQL are based on relational algebra and relational calculus.

In this chapter we will discuss relational algebra first and then relational calculus.

3.2 Relational Algebra

GTU : May-11,12, Winter-12,15,17,18, March-10, Summer-14,17, Marks 16

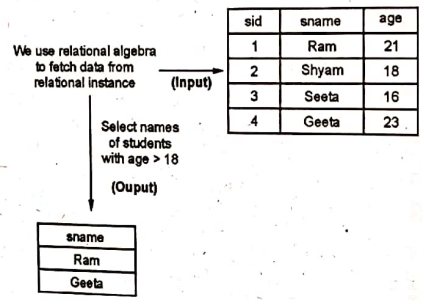
Definition :

- Relational algebra is a procedural query language which is used to access database tables to retrieve the information in the form of relational algebra expressions.
- The queries present in the relational algebra are denoted **using operators.**

Properties of Relational Algebra

(1) It makes use of operators and use relations(means tables) as operands.

(2) In relational algebra, one or two relations are used as input and the single relation is generated as output without name. For example :



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

sname
Ram
Geeta

(3) 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.

(4) The relational algebra is based on set theory and it does not allow duplicate entries.

(5) It does not make use of any English statements.

Importance of Relational Algebra

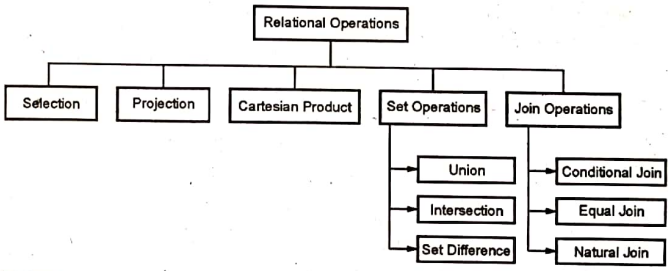
(1) It provides **formal foundation** for relational model operations

(2) It is used as a basis for **implementing and optimizing queries** in query processing.

(3) Some of the concepts of relational algebra are **incorporated into the SQL.**

3.2.1 Operations in Relational Algebra

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 σ represents 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.3.2.1 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 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. 3.2.2.
  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. 3.2.2

(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 × 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.

(1) 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 : A ∪ B**
  where A and B are relations.
- **For example :** If there are two tables student and book 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

$$\Pi_{Sname}(Student) \cup \Pi_{bname}(Book)$$

(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

$$\Pi_{name}(Student) \cap \Pi_{name}(Worker)$$

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 ⋈

There are three types of joins used in relational algebra

i) Theta 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 -

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

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 -

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

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

$(\sigma_{bname} = \text{'DBMS'} (\text{Book} \bowtie_{(Book.isbn = Reserve.isbn)} \text{Reserve})$

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 or sometimes * is used. 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	007	03-03-17
3	006	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

OR

Books * Reserver

The result of above two tables on natural join will be

Isbn	bname	Author	sid	day
005	DBMS	XYZ	1	07-07-18
006	OS	PQR	3	08-11-16
007	DAA	ABC	2	03-03-17

(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 :

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

(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.

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.

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.

Example 3.2.1 *Consider following databases reserves(sid, bid, day) sailors (sid, sname, rating, age) boats (bid,bname,color)*

i) Find the names of sailors who have reserved boat number 103

(ii) Find the names of sailors who have reserved a red boat

(iii) Find the id of sailors with age over 20 who have not reserved red boat

(iv) Find the names of sailors who have reserved at least one boat

Solution :

(i) $(\Pi_{sname}((\sigma_{bid=103}$ Reserves) ⋈ Sailors)

(ii) $(\Pi_{sname}((\sigma_{color='red'}$ Boats) ⋈ Reserves ⋈ Sailors)

(iii) $(\Pi_{sid}((\sigma_{age>20}$ Sailors) $- \Pi_{sid}((\sigma_{color='red'}$ Boats) ⋈ Reserves)

(iv) $(\Pi_{sname}$(Sailors ⋈ Reserves)

Example 3.2.2 *Consider the following expressions, which use the result of a relational algebra operation as the input to another operation. For each expression explain in words what the expression does :*

a) $\sigma_{year \geq 2009(takes)} \bowtie_{Student}$ *b)* $\sigma_{year \geq 2009(takes} \bowtie_{Student})$ *c)* $\Pi_{ID,\ name,\ course\text{-}id}$ *(student ⋈ takes)*

Solution :

a. Select each student who takes at least one course in 2009, display the student information along with the information about what the courses the student took.

b. Select each student who takes at least one course in 2009, display the student information along with the information about what the courses the student took but the selection must be before join operation.

c. Display the ID, Name and Course_id of all the students who took any course in the university.

Example 3.2.3 *Consider following relational database*
branch(branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
loan (loan_number, branch_name, amount)
borrower (customer_name, loan_number)
account (account_number, branch_name, balance)
depositor (customer_name, account_number)
i) Find the names of all branches located in "Chennai".
ii) Find the names of all borrowers who have a loan in branch "ABC".

Solution :

i) $\Pi_{branch\_name}(\sigma_{branch\_city = 'Chennai'})$ (branch))

ii) $\Pi_{customer\_name}(\sigma_{branch\_name = 'ABC'})$ (borrower ⋈ loan))

Example 3.2.4 *author (author_id, first_name, last_name)*
author_pub(author_id, pub_id, author_position)
book(book_id, book_title, month, year, editor)
pub(pub_id, title, book_id)

(i) Give the relational algebra expression that returns names of all the authors that are book editors
(ii) Give the relational algebra expression that returns names of all the authors that are not book editors
(iii) Write a relational algebra expression that returns the names of all authors who have at least one publication in the database.

Solution :

i) $(\Pi_{first\_name,last\_name}$(author $_{author\_id\ =\ editor} \bowtie$ book)

ii) $(\Pi_{first\_name,last\_name}((\Pi_{author\_id}$ (author) $- \Pi_{editor}$ (book)) × author)

iii) $(\Pi_{first\_name,last\_name}$ (author × author_pub)

Example 3.2.5 *Consider the following schema :*
Supplier(sid, sname,address)
Parts(pid, pname, color)
Cataloge(sid,pid,cost)
Write the relational algebraic queries for the following :
i) Find the sids of supplier who supply some red or some green parts
ii) Find the sids of supplier who supply every red or some green parts
iii) Find the pids of parts supplied by at least two different suppliers.

Solution :

i) $\rho(R1, \Pi_{sid}((\Pi_{pid}\ \sigma_{color='red'}$ Parts) ⋈ Cataloge))
$\rho(R2, \Pi_{sid}((\Pi_{pid}\ \sigma_{color='red'}$ Parts) ⋈ Cataloge))
R1 ∪ R2

ii) $\rho(R1, \Pi_{sid,\ pid}$Cataloge)/ $(\Pi_{pid}\ \sigma_{color='red'}$ Parts))
$\rho(R2, \Pi_{sid}((\Pi_{pid}\ \sigma_{color='red'}$ Parts) ⋈ Cataloge))
R1 ∪ R2

iii) ρ(R1, Cataloge)
ρ(R2, Cataloge)
$(\Pi_{R1.pid}\ \sigma_{R1.pid\ =\ R2.pid\ \wedge\ R1.sid!=R2.sid}$ (R1× R2)

Example 3.2.6 *Consider the relational database*
employee (person-name, street, city)
works (person-name, company-name, salary)
company (company-name, city)
manages (person-name, manager-name)
where primary keys are underlined.
(a) Find the names of all employees who work for First Bank Corporation

(b) Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than 200,000 per annum.

(c) Find the names of all employees in this database who live in the same city as the company for which they work.

Solution :

a) $\Pi_{\text{person-name}}(\sigma_{\text{company-name = "First Bank Corporation"}}(\text{works}))$

b) $\Pi_{\text{person-name, street, city}}(\sigma_{\text{company-name = "First Bank Corporation"} \wedge \text{salary} > 200000}(\text{works} \bowtie \text{employee}))$

c) $\Pi_{\text{person-name}}(\text{works} \bowtie \text{employee} \bowtie \text{company}))$

Example 3.2.7 *Solve the queries for the following database using relational algebra*

branch (branch-name, branch-city, assets)
customer (customer-name, customer-street, customer-only)
account (account-number, branch-name, balance)
loan (loan-number, branch-name, amount)
depositor (customer-name, account-number)
borrower (customer-name, loan-number)

(1) Find all loans over $1200
(2) Find the loan number for each loan of an amount greater than $1200
(3) Find the names of all customers who have a loan, an account, or both, from the bank
(4) Find the names of all customers who have a loan and an account at bank.
(5) Find the names of all customers who have a loan at the Perryridge branch.
(6) Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.
(7) Find the names of all customers who have a loan and an account at the Perryridge branch.

GTU : Winter-15, Marks 7

Solution :

(1) $\sigma_{\text{amount}>1200}(\text{loan}))$

(2) $\Pi_{\text{loan-number}}(\sigma_{\text{amount}>1200}(\text{loan}))$

(3) $\Pi_{\text{customer-name}}(\text{borrower}) \cup \Pi_{\text{customer-name}}(\text{depositor})$

(4) $\Pi_{\text{customer-name}}(\text{borrower}) \cap \Pi_{\text{customer-name}}(\text{depositor})$

(5) $\Pi_{\text{customer-name}}(\sigma_{\text{branch-name="Perryridge"}}(\sigma_{\text{borrower.loan-number=loan.loan-number}}(\text{borrower} \bowtie \text{loan})))$

(6) $\Pi_{\text{customer-name}}(\sigma_{\text{branch-name="Perryridge"}}(\sigma_{\text{borrower.loan-number=loan.loan-number}}(\text{borrower} \bowtie \text{loan})))$
$- \Pi_{\text{customer-name}}(\text{depositor})$

(7) $\Pi_{\text{customer-name}}(\sigma_{\text{branch-name="Perryridge"}}(\sigma_{\text{borrower.loan-number=loan.loan-number}}(\text{borrower} \bowtie \text{loan}))$
$\cup \Pi_{\text{customer-name}}(\text{depositor}))$

Example 3.2.8 *Consider the relational database as given below. Give an expression in relational algebra to express each of the following queries*

Employee (person-name, street, city)
Works (person-name, company-name, salary)
Company (company-name, city)
Manages (person-name, manager-name)

(a) Find the names of all employees in this database who live in the same city as the company for which they work.

(b) Find the names, street address, and cities of residence of all employees who work for HCL Corporation and earn more than $10,000 per annum.

GTU : Winter-17, Marks 4

Solution :

(1) $\Pi_{\text{person-name}}(\text{Works} \bowtie \text{Employee} \bowtie \text{Company}))$

(2) $\Pi_{\text{person-name, street, city}}(\sigma_{\text{company-name = "HCL"} \wedge \text{salary} > 10000}(\text{Works} \bowtie \text{Employee}))$

3.2.2 Additional Relational Operations

3.2.2.1 Generalized Projection

The generalized projection operation extends the projection operation by allowing functions of attributes to be included in the projection list.

The general form of generalized projection operation is -

$\Pi_{F1,F2,\ldots,Fn}(R)$

Where F1,F2,…,Fn are functions over the attributes in relation R and may involve arithmetic operations and constant values.

For example – Consider following relation for **Student** as

Student(RollNo,Name,Marks1,Marks2,Marks3)

Student

RollNo	Name	Marks1	Marks2	Marks3
1	AAA	40	50	60
2	BBB	55	56	57
3	CCC	70	80	90

Now if we want to have MarkSheet that need to show

Total = Marks1+Marks2+Marks3

Then generalized projection combined with renaming may be used as follows -

MARKSHEET $\leftarrow \rho_{(\text{RollNo,Name,Total})}(\Pi_{\text{RollNo,Name,Marks1+Marks2+Marks3}}(\text{STUDENT}))$

3.2.2.2 Aggregate Functions

The aggregate functions are the functions where values of multiple rows are grouped together as input on certain criteria to form a single value. For example finding average, or finding total number of employee, - such operations need to use aggregate functions.

Various aggregate functions are

1. Count()
2. Sum()
3. Avg()
4. Min()
5. Max()

The aggregate function is denoted by using the operator $\mathfrak{F}$ It ie pronounced as script F. The syntax of using aggregate function is -

$_{\text{<grouping attributes>}}\mathfrak{F}_{\text{<function list>}}(R)$

Where **<grouping attributes>** is a list of attributes of the relation R and **<function list>** is a list of (<function> and <attribute>). In this case <function> denotes any of the aggregate function and <attribute> denote the attribute of relation R.

Let us discuss these aggregate functions one by one

Suppose the relation Student as follows -

RollNo	Name	Marks
1	AAA	40
2	BBB	55
3	CCC	70
4	DDD	55
5	EEE	NULL

(1) Count : It will return total number of records

$\mathfrak{F}_{\text{COUNT RollNo}}$(STUDENT)

Will return 5

(2)Sum : It will return the sum of the values present in the attribute. For example

$\mathfrak{F}_{\text{SUM marks}}$(STUDENT)

Will return

220

(3) AVERAGE : It will return average value. For example

$\mathfrak{F}_{\text{AVERAGE marks}}$(STUDENT)

It will return

44

(4) MAXIMUM : It will return maximum value present in particular attribute. For example -

$\mathfrak{F}_{\text{MAXIMUM marks}}$(STUDENT)

It will return

70

(5) MINIMUM : It will return minimum value present in particular attribute. For example -

$\mathfrak{F}_{\text{MINIMUM marks}}$(STUDENT)

It will return

40

Example 3.2.9 *Consider following Schema and represent given statements in relation algebra form*

**Branch(branch_name, branch_city)*

**Account(branch_name, acc_no, balance)*

**Depositor(customer_name, acc_no)*

(i) Find out list of customer who have account at 'abc' branch

(ii) Find out all customer who have account in 'Ahemedabad' city and balance is greater than 10,000

(iii) Find out list of all branch name with their maximum balance

GTU : March-10, Marks 6

Solution :

(i) $\Pi_{\text{customer\_name}}(\sigma_{\text{branch\_name='abc'}}(\text{Account} \bowtie \text{Depositor}))$

(ii) $\Pi_{\text{customer\_name}}(\sigma_{\text{branch\_city='ahemedabad'} \wedge \text{balanace}>10000}(\text{Branch} \bowtie \text{Account} \bowtie \text{Depositor}))$

(iii) branch_name $\mathfrak{F}_{\text{max(balance)}}$(Account)

Example 3.2.10 *Consider the following relational database, where the primary keys are underlined. Give an expression in the relational algebra to express each of the following queries :*

employee(ssn,name,dno,salary,hobby,gender)

department(dno,dname,budget,location,mgrssn)

works_on(ssn,pno)

project(pno,pname,budget,location,goal)

1) List all pairs of employee names and the project numbers they work on

2) List out department number, department name and department budget

3) List all projects that Raj Yadav works on by project name

4) List the names of employee who supervise themselves

GTU : Summer-17, Marks 4

Solution :

(1) $\Pi_{name,pno}$(employee ⋈ works_on)

(2) $\Pi_{dno,dname,budget}$(department)

(3) RajYadav ← $\sigma_{name='Raj\ Yadav'}$(employee)

Result ← Π_{pname}(RajYadav ⋈ works_on ⋈ project))

(4) Insufficient data

Example 3.2.11 *Consider following relational database, where the primary keys are underlined. Give an expression in the relational algebra to express each of the following queries :*

course(course-id, title, dept_name, credits)

instructor(id,name,dept_name,salary)

section(course-id,sec-id,semester,year,building,room-no,time_slot_id)

teaches(id, course-id, sec-id, semester, year)

1. Find the names of all instructors in the physics department.

2. Find all the courses taught in the fall 2009 semester but not in Spring semester

3. Find the names of all instructors in Com.Sci.department together with course titles of all the courses that the instructors teach.

4. Find the average salary in each department.

GTU : Summer-17, Marks 4

Solution: (1) $\Pi_{name}(\sigma_{dept\_name='physics'}$(instructor))

(2) $\Pi_{course\text{-}id}(\sigma_{semester='Fall' \wedge year=2009}$ (section)) - $\Pi_{course\text{-}id}(\sigma_{semester='Sprint'}$ (section))

(3) $\Pi_{name,title}(\sigma_{dept\_name='Comp.Sci.department'}$(course ⋈ instructor ⋈ teaches))

(4) dept_name $\Im_{avg(salary)}$(instructor)

3.2.2.3 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

All rows from left table and only matching rows from right table

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

Student

RollNo	Name
1	AAA
2	BBB
3	CCC

Course

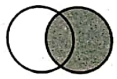
RollNo	CourseName
1	Computer
3	Electrical
5	Civil
6	Mechanical

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



All rows from right table and only matching rows from right table

- **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

All rows from both tables

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	$A \bowtie_{x=y} B$

Review Questions

1. *List relational algebra operators and explain any two with example* **GTU : May-11, Marks 5**
2. *What is relational algebra? Define relational algebra operation cross product with example* **GTU : May-12, Marks 3**
3. *Explain the select, project, natural join union and Cartesian product operations* **GTU : Winter-12, Marks 7**
4. *Write short note on relational algebra.* **GTU : Summer-14, Marks 7**
5. *Explain different types of outer join with example.* **GTU : Summer , Winter-18, Marks 7**

3.3 Relational Calculus

- Relational calculus in **non-procedural query language** and has no description about how the query will work or the data will be fetched. It only focusses on what to do, and not on how to do it.
- Relational calculus exists in two forms :
  - Tuple Relational Calculus (TRC)
  - Domain Relational Calculus (DRC)

3.4 Tuple Relational Calculus

- The Tuple Relational Calculus(TRC) is a non-procedural query language.
- In this relational calculus we have to find tuples for which a **predicate** is true.
- The calculus is dependent on the use of tuple variables. A tuple variable is a variable that 'ranges over' a named relation.
- **Syntax of TRC**

  {T|Formula(T)}

  In this form of relational calculus, we define a tuple variable T, which satisfies Formula (T).
- The Formula(T) can be any condition with reference to relation. For example {T| S ∈Students ∧ S.sname='Ram'} is a Tuple Relational Calculus,where variable S belongs to **Students** table and the name of the student is **Ram.**
- The formula is usually specified using **atomic formula**. The atomic formula is defined by following rules -
  - R ∈Relation.

  For example - the variable S ∈ Students table, or R∈ Reserve table.

- R.a op S.b where **op** is an operator.

For example - S.sid=R.sid

- R.a op constant

For example -S.age > 18

- **Rules for TRC :** Hence the formula can be recursively defined using one of the following things -
  - Any atomic formula
  - ¬p, p∧q, p∨q or p⇒q
  - There exists quantifier denoted by ∃. Hence ∃ R (Formula(R)) can be a formula
  - For all quantifier denoted by ∀. Hence ∀R (Formula(R)) can be a formula.
- The variable T defined in Tuple Relational Formula should always be a **free variable.**
- A free variable is a variable which is not denoted using the quantifiers ∃ and ∀. All the variables that are associated with ∃ and ∀ are called **bound variables.**
- Example of TRC Query : Consider two tables namely Student and Reserve.

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 id of students, names of students whose age >18. Then we can write it in TRC form as

{T|∃S(S∈ Student ∧ S.age > 18∧ T.sid = S.sid ∧ T.sname = S.sname)}

Query : Find the names of the students who reserved the book of isbn = 005

{T|∃ S∈ Student,∃ R ∈ Reserve (R.isbn = 005 ∧ S.sid = R.sid ∧ T.sname = S.sname)}

3.5 Domain Relational Calculus

- In Domain Relational Calculus (DRC), filtering is done based on the domain of the attributes and not based on the tuple values.
- **Syntax of DRC :** { c1, c2, c3, ..., cn | F(c1, c2, c3, ... ,cn)}

  where, c1, c2... etc represents domain of attributes(columns) and F defines the formula including the condition for fetching the data.
- The formula is usually specified using atomic formula. The atomic formula is defined by following rules -
  - R ∈ Relation.

For example : the variable S ∈ Students table,or R∈ Reserve table.

  - R.a op S.b where **op** is an operator.

For example - S.sid = R.sid

  - R.a op constant

For example - S.age > 18

- **Rules for DRC :** Hence the formula can be recursively defined using one of the following things :
  - Any atomic formula
  - ¬p, p∧q, p∨q or p⇒q
  - There exists quantifier denoted by ∃. Hence ∃R(F(X)) can be a formula where X is a domain variable.
  - For all quantifier denoted by ∀. Hence ∀R(F(X)) can be a formula where X is a domain variable.
- **Example :** Consider two tables namely **Student** and **Reserve**

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 all students with age >18

{<I,N,A>|< I, N, A > ∈ Student ∧ A > 18 }

Here I, N, A are the variable names for attributes of the relation Student. Hence < I, N, A > ∈ Student ensures that the domain variables I(for sid), N(for sname), and A(for age) are restricted to be fields of Student table.

Query : Find the names of the students who reserved the book of isbn = 005

{<N>|∃ I,A,(<I,N,A>∈ Student∧∃Lr, Br, D(< Ir,Br, > D∈ Reserve ∧ Ir = I ∧ Br = 005))}

3.6 SQL3

The SQL3 is a fourth revision of SQL database query language.

The commonly used Structured Query Language(SQL) standard is defined in 1992. It is commonly refred as SQL2 or SQL-92. The SQL2 does not support recursive queries. Hence we need to write PL/SQL or embedded SQL.

Later, on the ANSI and ISO SQL standardization added the features to the SQL specification in 1999 to support object-oriented data management. The SQL: 1999 is often referred to as SQL3 standard. SQL3 supports many of the complex data types.

The SQL3 supports recursive queries.

Various parts of SQL3 are -

(1) SQL/Framework

(2) SQL/Foundation

(3) SQL/CLI : An updated definition of the extension Call Level Interface.

(4) SQL/PSM : An updated definition of the extension Persistent Stored Modules.

(6) SQL/Bindings :

Three more parts, also considered.

(7) SQL/MED Management of External Data

(8) SQL/OLB Object Language Bindings

(9) SQL/JRT SQL Routines and Types using the Java Programming Language.

Features of SQL3

(1) It has a support for relational and object oriented model

(2) It contains rich set of new data types such as LARGE OBJECT, CHARACTER LARGE OBJECT,BINARY LARGE OBJECT.

(3) It contains new Predicates such as DISTINCT

(4) It allows to use recursive queries

(4) It allows to use recursive queries.

(5) It has enhanced security

(6) It has a support for active databases using triggers.

(7) It contains allows to use structured user defined data types. For example -

create type age as integer FINAL;

create type salary as integer FINAL;

(8) It allows to use references and pointers.

3.7 DDL and DML Constructs

(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. For that purpose the SELECT command is used.
  - **Insertion** of new information into the database. For that purpose INSERT command is used.
  - **Deletion** of information from the database. For that purpose DELETE command is used.
  - **Modification** of information stored in database. For that purpose UPDATE command is used.

3.8 Open Source and Commercial DBMS

There are two types of database management software - Open source DBMS software and Commercial DBMS software. The commercial product is a paid software while open source DBMS is freely available on Internet.

Let us understand the difference between the two

Difference between Open Source and Commercial DBMS

Sr.No.	Open Source DBMS	Commercial DBMS
1.	The open source DBMS products are offered free of cost.	These DBMS products are paid software products. The cost can vary depending upon the complexity of the product.
2.	The code of open source DBMS product can be viewed, shared or modified by the community. That means any one can fix, upgrade or test the broken code.	The commercial DBMS products can be fixed only by vendors.
3.	As the code can be fixed by any one, it can be done efficiently.	As the code fixing is vendor specific, it may take some time.
4.	There are chances of malfunctioning with the code as source code is open.	The security is high and code is not at accessible to unauthorized user.
5.	When the support is required, there are various forums available to help the developers.	When the support is required, you need to contact vendor.
6.	Documentation is available for development point of view. So it might be complex and not suitable for layperson user.	Documentation is available from user's perspective. Hence it is well-structured, with clear and well-written instructions.
7.	**Examples -** MySQL, PostgreSQL, Oracle	**Examples -** Microsoft SQL Server

3.8.1 MYSQL

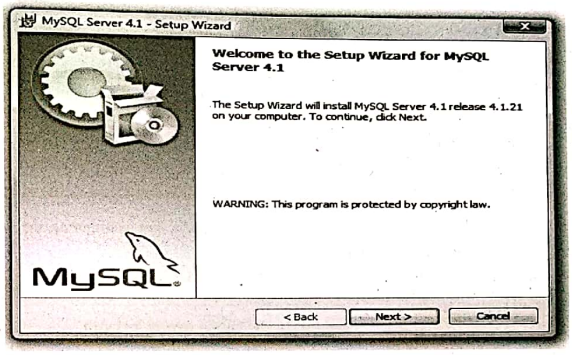
MYSQL is a open source database product and can be downloaded from the website **http://dev.mysql.com/downloads/mysql.**

MYSQL is a kind of database in which the records are stored in an entity called **tables.** In the tables the data is arranged in the rows and columns. We can query a database to retrieve particular information. **Query** is a request or a question for the database. There is a common practice of making use of Structured Query Language(SQL).

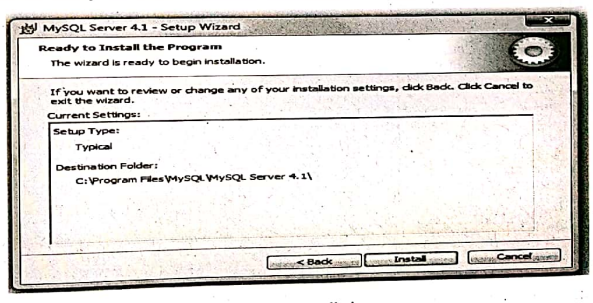
In this section we will first see the process of installing MYSQL and then we will discuss some commonly used queries.

3.8.1.1 Installation of MYSQL

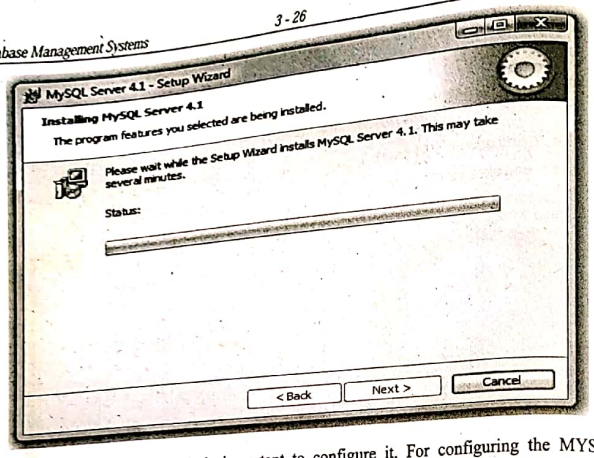
I have downloaded the MYSQL server for Windows which is there in the ZIP format. Extract the contents and execute the **setup.exe**. Following window may appear on the screen -



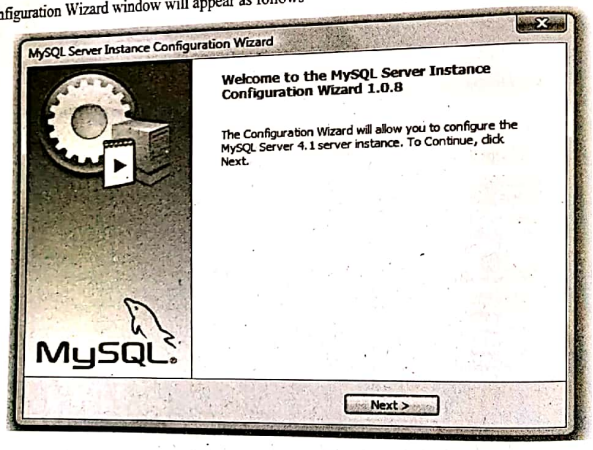
Click **Next**. It will then display the default directory in which the MYSQL server software will get installed. Accept this default location by clicking **Install** button.



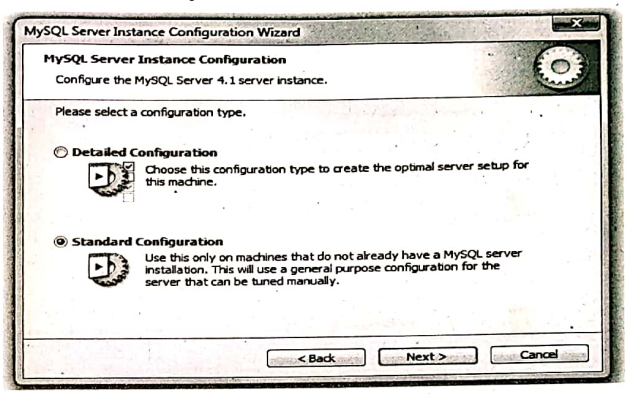
The setup wizard will display the progress of installation.



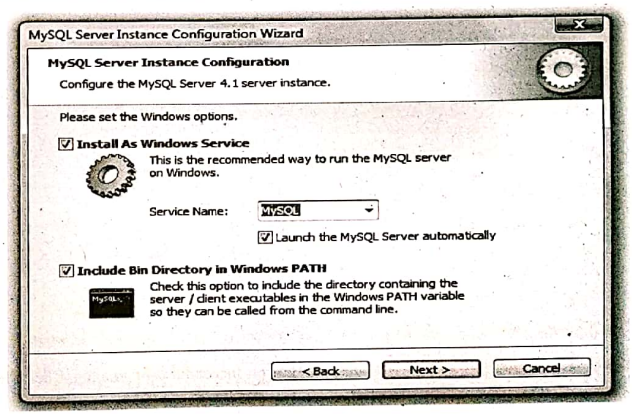
After installing the software, it is important to configure it. For configuring the MYSQL configuration Wizard window will appear as follows -



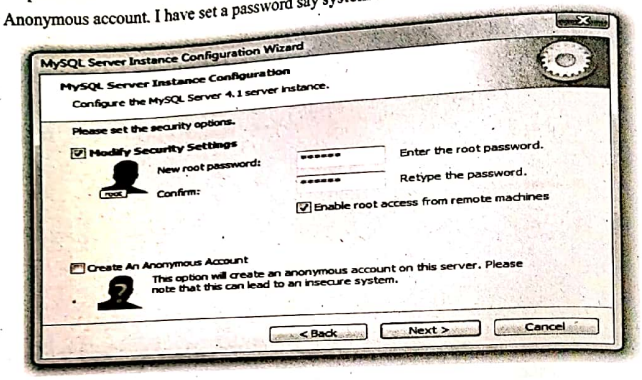
Just click the **Next** button to proceed. Select the **Standard Configuration** option.



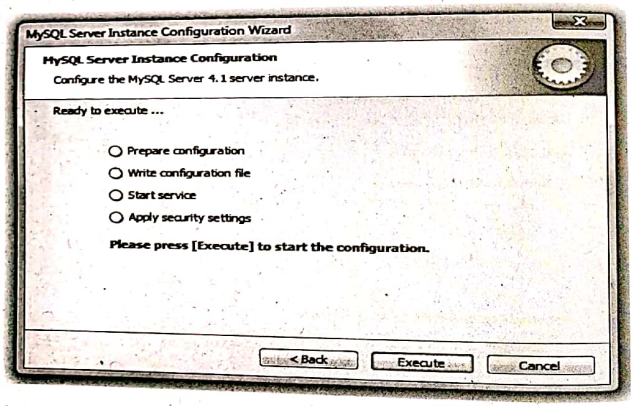
Then click Next button and following window will appear which allows you to set the Service Name. Just check the box before **Include Bin Directory in Windows PATH** so that the environmental variables can be set.



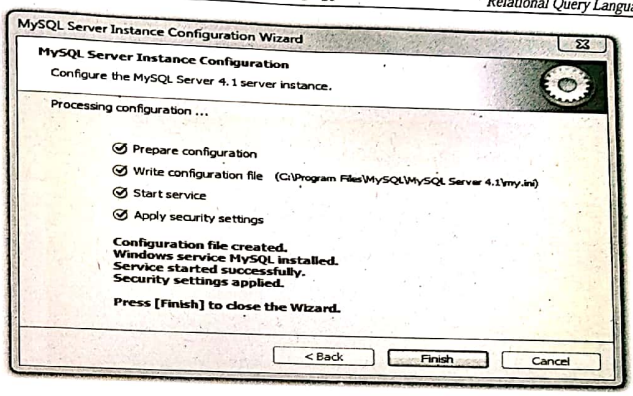
Then following window will appear which allows you to set the password for your SQL server. This option is for security purpose. If you don't want to set any password you can click on create an Anonymous account. I have set a password say **system** and then click **Next** button.



Now MYSQL server gets configured. Press Execute button.

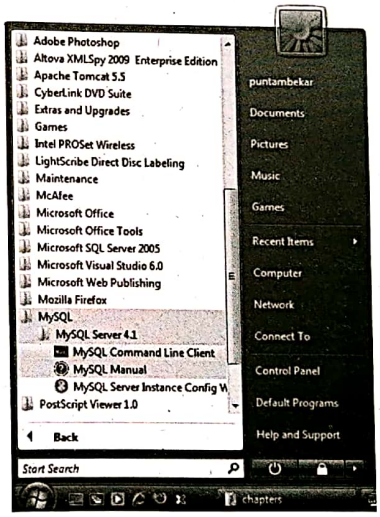


After some moments you will get following window on successful configuration. Click **Finish** button. There is one configuration file **my.inf** for MYSQL in which the information about the configuration is stored.

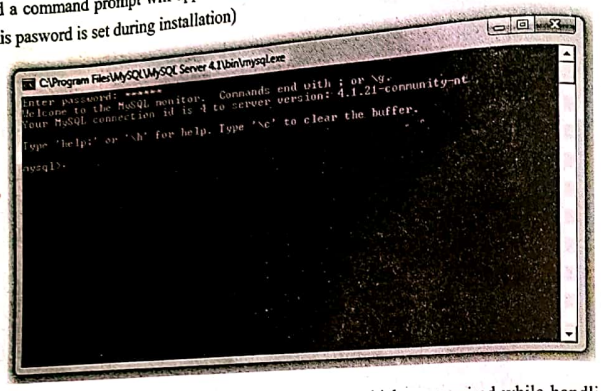


Now to start SQL server go to start->All Programs->MYSQL->MYSQL Server 4.1

Select the MYSQL Command Line Client



and a command prompt will appear after you type the password system. It is as shown (Note that this pasword is set during installation)



Now let us go through some MYSQL query statements which are required while handling the database.

3.8.1.2 Executing Queries

1. Creating database

```
mysql> CREATE DATABASE mydb;
Query OK, 1 row affected (0.15 sec)
```

2. Displaying all the databases

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mydb     |
| mysql    |
| students |
| test     |
+----------+
4 rows in set (0.06 sec)
```

3. Selecting particular database

```
mysql> USE MYDB;
Database changed
```


4. Creating table

We must create a table inside a database hence it is a common practice to use create table command after USE database command. While creating a table we must specify the table fields.

```
mysql> CREATE TABLE my_table(id INT(4),name VARCHAR(20));
Query OK, 0 rows affected (0.28 sec)
```

5. Displaying a table

After creating the table using SHOW command we can see all the existing tables in the current database.

```
mysql> SHOW TABLES;
+----------------------+
| Tables_in_mydb       |
+----------------------+
| my_table             |
+----------------------+
1 row in set (0.00 sec)
```

6. Displaying the table fields

For knowing the various fields of the table we may use following command

```
mysql> DESCRIBE my_table;
+---------+---------------+-------+-----+---------+--------+
| Field   | Type          | Null  | Key | Default | Extra  |
+---------+---------------+-------+-----+---------+--------+
| id      | int(4)        | YES   |     | NULL    |        |
| name    | varchar(20)   | YES   |     | NULL    |        |
+---------+---------------+-------+-----+---------+--------+
2 rows in set (0.07 sec)
```

7. Inserting values into the table

We can insert only one complete record at a time. It is as shown below -

```
mysql> INSERT INTO my_table
   -> VALUES(1,'SHILPA');
Query OK, 1 row affected (0.05 sec)
```

8. Displaying the contents of the table

```
mysql> SELECT * FROM my_table;
+------+-----------+
|  id  | name      |
+------+-----------+
|   1  | SHILPA    |
+------+-----------+
1 row in set (0.06 sec)
```

We can also write SELECT statement for selecting particular row by specifying some condition such as -

```
mysql> SELECT * FROM my_table where id=1;
or
mysql> SELECT * FROM my_table where name='SHILPA';
```

Thus we can insert the rows into the table by repeatedly giving the INSERT command.

If we want to get the records in sorted manner then we use ORDER BY clause

```
mysql> SELECT * FROM my_table;
+----+----------+
| id | name     |
+----+----------+
|  1 | SHILPA   |
|  2 | SUPRIYA  |
|  3 | YOGESH   |
|  4 | MONIKA   |
+----+----------+
4 rows in set (0.00 sec)
mysql> SELECT * FROM my_table ORDER BY name;
+----+----------+
| id | name     |
+----+----------+
|  4 | MONIKA   |
|  1 | SHILPA   |
|  2 | SUPRIYA  |
|  3 | YOGESH   |
+----+----------+
4 rows in set (0.00 sec)
```

9. Updating the record

For updating the record in the database following command can be used -

```
mysql> UPDATE my_table
  -> SET name='PRIYANKA'
  -> WHERE id=4;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> SELECT * FROM my_table;
+----+----------+
| id | name     |
+----+----------+
|  1 | SHILPA   |
|  2 | SUPRIYA  |
|  3 | YOGESH   |
|  4 | PRIYANKA |
+----+----------+
4 rows in set (0.00 sec)
```


10. Deleting record

For deleting particular record from a database

```
mysql> DELETE FROM my_table
  -> WHERE id=3;
Query OK, 1 row affected (0.04 sec)
```

Then use SELECT statement for displaying the contents of the table we use following command

```
mysql> SELECT * FROM my_table;
+----+----------+
| id | name     |
+----+----------+
|  1 | SHILPA   |
|  2 | SUPRIYA  |
|  4 | PRIYANKA |
+----+----------+
3 rows in set (0.00 sec)
```

11. For deleting the table

The table can be deleted using the command

```
mysql>drop table my_table;
```

3.8.2 ORACLE

Oracle is a relational database management system. It is known as oracle DB or just oracle.

The oracle DB is produced by Oracle Corporation.

Features of Oracle

(1) Availability : Oracle allows read-only access to data in physical standby database so that user can perform various tasks such as reporting, querying, some data extraction and backup.

(2) Scalability : There is an availability of Oracle Real Application cluster which provides the facility for the database for application continuity, online instance migration, and quality of service management.

(3) Performance : Oracle provides **improved** performance of **transaction processing.**

(4) Security : Oracle has a sophisticated and flexible framework for **fine access control** implementation which support strong security feature.

(5) Tools : Oracle SQL Developer, a free graphical tool for database development, allows developer to browse database objects, to run SQL statements and SQL scripts, and to edit and

debug PL/SQL statements. It incorporates standard and customized reporting. Users can also develop their own applications in Java and PL/SQL using the tools such as -

(i) Oracle Forms

(ii) Oracle JDeveloper

(iii) Oracle Reports.

Editions of Oracle Database

There are four editions of Oracle database –

- **Enterprise Edition :** It is the most robust and secure edition. It offers all features, including superior performance and security.
- **Standard Edition :** It provides the base functionality for users that do not require Enterprise Edition's robust package.
- **Express Edition (XE) :** It is the lightweight, free and limited Windows and Linux edition.
- **Oracle Lite:** It is designed for mobile devices.

3.8.2.1 Commands for using Oracle Database

We can use various commands to create database, tables and to perform various operations on table.

1. Creation of database

Syntax :

```
CREATE DATABASE database_name
```

Example :

```
CREATE DATABASE test
```

2. Create Tables

Using CREATE TABLE command we can create a table in Oracle.

Syntax :

```
CREATE TABLE table_name
(
   Column1 datatype [NULL | NOT NULL]
   Column2 datatype [ NULL | NOT NULL ],
   ...
   Column_n datatype [ NULL | NOT NULL ]
);
```


For Example :

```
CREATE TABLE Students
(Roll number(10) NOT NULL,
name  varchar2(30) NOT NULL,
Marks number(5)
);
```

We can also set the primary key while creating the table.

For example

```
CREATE TABLE Students
(Roll number(10) NOT NULL,
name  varchar2(30) NOT NULL,
Marks number(5)
CONSTRAINT stud_id PRIMARY KEY(Roll)
);
```

3. Insert Data into Table

Using the insert statement we can insert some values to the table.

Syntax :

```
INSERT INTO table
(column1, column2, ... column_n )
VALUES
(expression1, expression2, ... expression_n );
```

Example :

```
INSERT INTO my_table(Roll,name,marks)
VALUES(1,'SHILPA',55);
```

Output

```
1 row(s) inserted.
0.02 seconds
```

4. Retrieve Information from Table

For retrieving the information from table we use SELECT statement.

Syntax :

```
SELECT expressions
FROM tables
WHERE conditions;
```

Example :

```
SELECT *
FROM Student;
```

The above query will display all the records present in the Student table.

We can also filter some records while displaying by mentioning the condition using WHERE clause. For example

```
SELECT Roll,name
FROM Student
WHERE marks>50;
```

Using above query only those students' Roll number and names will be displayed whose marks >50

5. Delete Data from Table

We can delete single record or multiple records from table.

Syntax :

```
DELETE FROM table_name
WHERE conditions;
```

For example :

```
DELETE FROM Students
WHERE name='Shilpa'
```

6. Update Data in Table

Using Update statement, we can change the record values.

Syntax :

```
UPDATE table_name
SET column1 = expression1,
column2 = expression2,
...
column_n = expression_n
WHERE conditions;
```

For example :

```
UPDATE Student
SET name='Rupali'
WHERE Roll=1;
```

Output

```
1 row(s) updated.
0.06 seconds
```

3.8.3 DB2

- DB2 is a database product developed by IBM.

- It is a relational database management system.
- It is used to perform basic database operations such as storing of data, deleting, modifying the data, retrieving the data.
- DB2 also supports object oriented features and non relational structure with XML.

Features

(1) It has more powerful Structured Query Language (SQL) dialect than Microsoft's SQL offering. DB2 has features such as object tables, before triggers, Java method support, multiple user-defined functions and support for arrays.

(2) It also can be imbedded in the code of application programs written in other languages, such as COBOL and Java.

(3) IBM produces versions of DB2 that run on all available platforms, rather than just Windows-based platforms such as AIX, HP-UX, Linux and Sun.

(4) It has self tuning memory management. During the day access is typically via online applications, with random access to records across the database. At night, the workload typically changes to a batch format, with sequential processing of records.

(5) It has a strong support from IBM by providing time to time updates and patches after thorough testing.

3.8.4 SQL Server

- SQL Server is a relational database management system developed by Microsoft.
- SQL Server supports ANSI SQL, which is the standard SQL (Structured Query Language) language. However, SQL Server comes with its own implementation of the SQL language, T-SQL (Transact-SQL).
- T-SQL is a Microsoft propriety Language known as Transact-SQL. It provides further capabilities of declaring variable, exception handling, stored procedure and so on.

Features

(1) **On Linux :** It can run on Windows as well as on Linux.

(2) **Machine Learning Services :** One can execute the Python script within the database server itself. Both R and Python are popular programming languages that provide extensive support for data analytics along with natural language processing capability.

(3) **Optimized Query Processing :** SQL Server 2017 adapts optimization strategies to your application workload's runtime conditions. It includes adaptive query processing features that you can use to improve query performance in SQL Server and SQL Database.

(4) Automatic Database Tuning : Whenever a potential performance issue is detected and enables you to apply corrective actions, or it enables the database engine to automatically fix performance issues.

(5) Smart Differential Backup and Transaction Log Backup : It provides support for smart backup system and transaction log backups by adding intelligence to backup solutions.

(6) Security Enhancements : You can now grant, deny, or revoke permissions on database-scoped credentials such as CONTROL, ALTER, REFERENCES, TAKE OWNERSHIP, and VIEW DEFINITION permissions.

Editions of SQL Server

Following editions are available

SQL Server Enterprise : It is used in the high end, large scale and mission Critical business. It provides High-end security, Advanced Analytics, Machine Learning, etc.

SQL Server Standard : It is suitable for Mid-Tier Application and Data marts. It includes basic reporting and analytics.

SQL Server WEB : It is designed for a low total-cost-of-ownership option for Web hosters. It provides scalability, affordability, and manageability capabilities for small to large scale Web properties.

SQL Server Developer : It is similar to an enterprise edition for the non-production environment. It is mainly used for build, test, and demo.

SQL Server Express : It is for small scale applications and free to use.

Review Questions

1. *What is the difference between commercial and Open source DBMS ?*
2. *Write short note on MySQL*
3. *What are the features of ORACLE database.*

3.9 Oral Questions with Answers

Q.1 Name two formal languages used in relational algebra.

Ans. : There are **two formal languages** used for relational model and those are –

- Relational Algebra
- Relational Calculus

Q.2 Define formally the term – Relational Algebra.

Ans. :

- Relational algebra is a procedural query language which is used to access database tables to retrieve the information in the form of relational algebra expressions.
- The queries present in the relational algebra are denoted **using operators.**

Q.3 What is importance of Relational Algebra ?

Ans. :

(1) It provides **formal foundation** for relational model operations.

(2) It is used as a basis for **implementing and optimizing queries** in query processing.

(3) Some of the concepts of relational algebra are **incorporated into the SQL.**

Q.4 What is the purpose of selection operation in relational algebra ?

Ans. : Selection operation is used to fetch the rows or tuples from the table(relation).

Q.5 What is the purpose of projection operation in relational algebra ?

Ans. :

- 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.
- In order to display particular column from the relation, the projection operator is used.

Q.6 List various operators used in relational algebra.

Ans. : Various operators used in Relational algebra are –

(1) Selection Operator(σ)

(2) Projection Operator(∏)

(3) Cartesian Product(×)

(4) Rename Operator(ρ)

Q.7 What are aggregate functions ?

Ans. : The aggregate functions are the functions where values of multiple rows are grouped together as input on certain criteria to form a single value. For example finding average, or finding total number of employee, - such operations need to use aggregate functions.

Q.8 List various aggregate functions used.

Ans. : Various aggregate functions are

1. Count()
2. Sum()
3. Avg()

4. Min()
5. Max()

Q.9 What is relational calculus ?

Ans. : Relational calculus in non-procedural query language and has no description about how the query will work or the data will be fetched. It only focusses on what to do, and not on how to do it.

Q.10 Enlist two forms of relational calculus.

Ans. : Relational calculus exists in two forms :

(1) Tuple Relational Calculus (TRC).

(2) Domain Relational Calculus (DRC).

Q.11 What is atomic formula ?

Ans. : The atomic formula is defined by following rules -

- R ∈ Relation.

For example - the variable S ∈ Students table, or R∈Reserve table.

- R.a op S.b where **op** is an operator.

For example - S.sid=R.sid

- R.a op constant

For example -S.age > 18

Q.12 What is domain relational calculus ?

Ans. :

- In Domain Relational Calculus (DRC), filtering is done based on the domain of the attributes and not based on the tuple values.
- **Syntax of DRC :** { c1, c2, c3, ..., cn | F(c1, c2, c3, ... ,cn)}

where, c1, c2... etc represents domain of attributes(columns) and F defines the formula including the condition for fetching the data.

Q.13 Enlist some features of SQL3.

Ans. :

(1) SQL3 has a support for relational and object oriented model.

(2) It contains rich set of new data types such as LARGE OBJECT, CHARACTER LARGE OBJECT,BINARY LARGE OBJECT.

(3) It contains new Predicates such as DISTINCT.

(4) It allows to use recursive queries.

❑❑❑

4 Relational Database Design

Syllabus

Domain and data dependency, Armstrong's axioms, Normal forms, Dependency preservation Lossless design.

Contents

4.1 Domain and Data Dependency

- There are two primary goals of relational database design – i) to generate a set of relation schemas that allows us **to store information without unnecessary redundancy,** and ii) to allows us to **retrieve information easily.**
- For achieving these goals, the database design need to be **normalized**. That means we have to check whether the schema is it normal form or not.
- For checking the normal form of the schema, it is necessary to check the Domain and data dependencies that exists within the schema.

Hence before letting us know what the normalization means, it is necessary to understand the concept of functional dependencies.

4.2 Functional Dependency

GTU : Dec.-05, 06, May-11, June-05, 06 , Winter-15, 17, Marks 8

Definition : A functional dependency A->B in a relation holds if two tuples having same value of attribute A also have the same value for attribute B. It is denoted by A->B where A is called determinant and B is called dependent.

For example - Consider Student table as follows –

Roll	Name	City
1	AAA	Mumbai
2	BBB	Pune
3	CCC	Gandhinagar

Here

Roll ->Name **hold**

But

Name->City **does not hold**

- In above table, Student Roll number is unique hence each student's name and city can be uniquely identified using his Roll number.
- But using name we cannot uniquely identify his/her city because there can be same names of the students. Similarly using city name we can not identify the student uniquely. As in the same city may belong to multiple students.

Another example –

Consider a relation in which the roll of the student and his/her name is stored as follows :

R	N
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 4.2.1 : Table which holds functional dependency i.e. R->B

Here, R->N is true. That means the functional dependency holds true here. Because for every assigned RollNuumber of student there will be unique name. For instance : The name of the Student whose RollNo is 1 is AAA. But if we get two different names for the same roll number then that means the table does not hold the functional dependency. Following is such table -

R	N
1	AAA
2	BBB
3	CCC
1	XXX
2	YYY

Fig. 4.2.2 : Table which does not hold functional dependency

In above table for RollNumber 1 we are getting two different names - "AAA" and "XXX". Hence here it does not hold the functional dependency.

Trivial FD : The functional dependency A->B is trivial if B is a subset of A. For example

{A,B}->A

Non Trivial FD : The functional dependency A->B is non trivial if B is not a subset of A.

For example {A,B}->C

4.2.1 Armstrong's Axioms

The closure set is a set of all functional dependencies implied by a given set F. It is denoted by F^+.

The closure set of functional dependency can be computed using basic three rules which are also called as Armstrong's Axioms.

These are as follows -

i) **Reflexivity :** If $X \supseteq Y$, then $X \rightarrow Y$

ii) **Augmentation :** If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z

iii) **Transitivity :** If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

In addition to above axioms some additional rules for computing closure set of functional dependency are as follows -

- **Union :** If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
- **Decomposition :** If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Let us understand how to apply Armstrong's axioms for finding the closure of set of functional dependencies –

Example 4.2.1 *Given FD's for relation R(A,B,C,D,E,F), Find closure of FD set by applying Armstrong's Axioms.* **GTU : June-05, Marks 6**

A->B, A->C, CD->E, CD->F, B->E

Solution :

Step 1 : A->gives A attribute itself by reflexivity. It is called trivial production.

A->B and A->C, Hence by union rule A->BC

A->B and B->E, Hence by transitivity rule A->E

$\therefore (A)^+ = \{A,B,C,E\}$

Step 2 : B-> gives B itself

And B->E

$\therefore (B)^+ = \{B,E\}$

Step 3 : CD->gives CD itself. It is trivial.

CD->E, CD->F, Hence by union rule CD->EF

$\therefore (CD)^+ = \{C,D,E,F\}$

So by omitting trivial productions, we get

$\therefore F^+ = \{A->BC, A->E, B->E, CD->EF\}$

Example 4.2.2 *Compute the closure of the following set F of functional dependencies for relational schema R=(A,B,C,D,E) A->BC, CD->E, B->D, E->A* **GTU : Dec.-05, Marks 2**

Solution : The closure of F is denoted by F^+ and it can be computed in following steps

Step 1 : As A->BC is given we get

A->B and A-> C — By decomposition rule

Step 2 : As

A->B — (Refer step 1)

B->D — (given)

A->D — (transitivity rule)

Step 3 :

A->CD because A->C and A->D — (From step 1 and Step 2 applying union rule)

Step 4 :

CD->E — (given)

A->E — (transitive rule as A->CD, CD->E hence A->E)

Step 5 :

Since A → A, — we have (reflexive)

A → ABCDE — from the above steps (union)

Step 6 :

Since E → A, **E → ABCDE** — (transitive)

Step 7 :

Since CD → E, **CD → ABCDE** — (transitive)

Step 8 :

Since B → D and BC → CD, **BC → ABCDE** (augmentative, transitive)

Step 9 :

Also, C → C, D → D, BD → D

Thus any functional dependency with A, E, BC, or CD on the left hand side of the arrow is in F^+

Example 4.2.3 *Give Armstrong's axioms and using it find the closure of following FD set.* **GTU : June-06, Marks 8**

A->B, AB->C, D->AC, D->E

Solution :

Step 1 : Consider the rules D->AC and D->E **(Union rule)**

∴ **D->ACE**

Step 2 : Consider AB->C then we get **(decomposition rule)**

∴ A->C and B->C

Thus F^+ ={A->C, B->C, D->ACE}

Example 4.2.4 *Here are two sets of FDs for R(A,B,C,D,E). Are they equivalent ?*

1) A->B AB->C D->AC D->E

2) A ->BC D->AE

GTU : Dec.-06, Marks 3

Solution : Two sets of FDs are said to be equivalent if

Rule 1 : If FD2 ⊃ FD1. That means all FDs of FD1 can be derived from all the FDs of FD2

Rule 2 : If FD1 ⊃ FD2. That means all FDs of FD2 can be derived from all the FDs of FD1

Rule 3 : If both Rule 1 and Rule 2 are true then FD1=FD2

For given two sets

Step 1 : We will first check if all the FDs of FD1 are present in FD2

A->B is present in FD1.	A->BC is in FD2, that also means A->B and A->C by decomposition rule
Similarly AB->C is in FD1	Hence
∴ $(A)^+$ = {A,B,C}	$(A)^+$ = {A,B,C}
∴ $(AB)^+$ = {A,B,C}	As D->AE then by decomposition rule,
D->AC	D->A, D->E
i.e D->A, D->C by decomposition rule.	As A->B, then by transitivity rule D->A, A->B, D->B
The D->A, A->B and C	∴ $(D)^+$ = {A,B,C,D,E}
D->A, D->B, D->C by transitivity rule	
D->E is given	
∴ $(D)^+$ = {A,B,C,D,E}	

Step 2 : We will first check if all the FDs of FD2 are present in FD1

∴ $(A)^+$ = {A,B,C}	$(A)^+$ = {A,B,C}
∴ $(AC)^+$ = {A,B,C}	∴ $(D)^+$ = {A,B,C,D,E}
∴ $(D)^+$ = {A,B,C,D,E}	

Thus from Step 1 and Step 2, FD2 ⊃ FD1 and FD1 ⊃ FD2. Hence both the sets are equivalent.

Example 4.2.5 *R = {A,B,C,D,E,F} and FDs are A->BC E->CF B->E CD->EF compute closure of {A,B}+*

GTU : Dec.-06, May-11, Winter-15, Marks 3

Solution :

Step 1 : A->BC

∴ A->B and A->C (decomposition)

So we add A, B, C in closure set

Step 2 : E->CF

∴ E->C and E->F (decomposition)

Step 3 : B->E ∴ B->C, F

So we add E and F in closure set

Hence

$\{A,B\}^+$ = {A,B,C,E,F}

Example 4.2.6 *Consider schema EMPLOYEE(E-ID, E-NAME, E-CITY, E-STATE) and FD = {E-ID->E-NAME, E-ID->E-CITY, E-ID->E-STATE, E-CITY-> E-STATE}*

(1) Find attribute of closure for(E-ID)+

(2) Find(E-NAME)+

GTU : Winter-17, Marks 4

Solution :

(1) Finding $(E\text{-}ID)^+$ means finding the closure. In this process we try to find out, all the attributes that can be derived from E-ID.

As E-ID->E-NAME, E-ID->E-CITY, E-ID->E-STATE, We add E-NAME, E-ID, E-CITY, E-STATE to $(E\text{-}ID)^+$

As E-ID->E-CITY, E-CITY->E-STATE, we add E-STATE to $(E\text{-}ID)^+$

∴ $(E\text{-}ID)^+$ = {E-ID, E-NAME, E-CITY, E-STATE)

(2) **E-NAME** derives no rule. Hence $(E\text{-}NAME)^+$ = {E-NAME}

4.2.2 Keys and Functional Dependencies

- For a given relation R={A1, A2, A3, ..., An} K is a key of R then if closure $(K)^+$ = {A1, A2, ..., An} and no subset of K i.e. X such that $(X)^+$ = A1, A2, ..., An}
- In other words there are two conditions –

  (1) The $(K)^+$ contains all the attributes of relations R

  (2) All subset X of K, $(X)^+$ never contains all the attributes of R i.e. $(X)^+ \neq$ {A1, A2, ..., An}
- If only one subset of R satisfy above condition then it is known as **primary key**

- If more than one subset of R satisfies above condition then all these subsets are recognized as **candidate keys.** In that case one of the candidate key is also considered as **primary key**
- A superset of candidate key K is known as **superkey**

Example 4.2.7 *Give R = {A,B,C,G,H,I}. The following set F of functional dependencies holds A->B, A->C, CG->H, CG->I, B->H*
Computer AG^+. Is AG candidate key ? **GTU : Winter-15, Marks 3**

Solution :

Step 1 : A->B, A->C, Hence add A, B, C to the set of AG^+.

Step 2 : A->B B-> H, hence add H to the set of AG^+

Step 3 : CG->I, hence add I to the set of AG^+. Also add G to the set.

Thus $(AG)^+$ = {A,B,C,G,H,I} = Relation R

Hence $(AG)^+$ is a candidate key.

Example 4.2.8 *Compute the closure of R(A,B,C,D,E) with the following set of functional dependencies*
A->BC, CD->E, B->D, E->A
List the candidate keys of R **GTU : Winter-15, Marks 3**

Solution :

Step 1 : A->BC hence add A,B,C to $(A)^+$

A->BC can be decomposed into A->B and A->C. Also B->D. Thus A->D is also true by transitivity rule.

Hence add D to $(A)^+$

A->C, A->D ∴ By union rule A->CD.

As CD->E add E to $(A)^+$

∴ $(A)^+$ = {A,B,C,D,E}

Step 2 : Consider $(B)^+$ = {B,D}≠ R hence it is not a candidate key

Step 3 : Consider $(BC)^+$ = {B,C,D,E,A}={A,B,C,D,E} = R. Hence it is a candidate key

Step 4 : Consider CD->E, E->A, hence $(CD)^+$ = {A,B,C,D,E}. Hence it is a candidate key

Step 5 : Consider E->A, A->BC, B->D, CD->E. Hence $(E)^+$ = {A,B,C,D,E} is a candidate key.

Thus we get the candidate keys as {A,BC, CD, E}

Example 4.2.9 *Consider Schema R = (A,B,C,G,H,I) and the set F of functional dependencies {A->B, A->C, CG->H, CG->I, B->H}. Use $(F)^+$ Prove $(AG)^+$->I* **GTU : Winter-17, Marks 3**

Solution :

Step 1 : As A->B
B->H
∴ **A->H** (transitivity rule)

Step 2 : CG->H
CG->I
∴ **CG->HI** (Union rule)

Step 3 : A->C
CG->I
AG->I (Pseudo transitive rule)

Thus AG->I is proved

$(F)^+$ = {A->B, A->C, CG->H, CG->I, B->H, A->H, CG->HI, AG->I}

4.2.3 Canonical Cover or Minimal Cover

Formal Definition : A minimal cover for a set F of FDs is a set G of FDs such that :

1) Every dependency in G is of the form X->A, where A is a single attribute.
2) **The closure F^+** is equal **to the closure G^+.**
3) If we obtain a set H of dependencies from G by deleting one or more dependencies or by deleting attributes from a dependency in G, then $F^+ \neq H^+$.

Concept of Extraneous Attributes

Definition : An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies. The formal definition of extraneous attributes is as follows :

Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F

- Attribute A is extraneous in α if $A \in \alpha$, and F logically implies
  $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$
- Attribute A is extraneous in β if $A \in \beta$ and the set of functional dependencies
  $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha \rightarrow (\beta - A)\}$ logically implies F.

Algorithm for computing Canonical Cover for set of functional Dependencies F

$F_c = F$

repeat

Use the union rule to replace any dependencies in F_c of the form

$\alpha 1 \rightarrow \beta 1$ and $\alpha 1 \rightarrow \beta 2$ and $\alpha 1 \rightarrow \beta 1\beta 2$

Find a functional dependency $\alpha \rightarrow \beta$ in F_c with an extraneous attribute either in α or in β.

/* The test for extraneous attributes is done using F_c, not F */

If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$ in F_c.

until (F_c does not change)

Example 4.2.10 *Consider the following functional dependencies over the attribute set R(ABCDE) for finding minimal cover FD = {A->C, AC->D, B->ADE}*

Solution :

Step 1 : Split the FD such that R.H.S contain single attribute. Hence we get

A->C
AC->D
B->A
B->D
B->E

Step 2 : Find the **redundant entries** and **delete** them. This can be done as follows -

- **For A->C :** We find $(A)^+$ by assuming that we delete A->C temporarily. We get $(A)^+$={A}. Thus from A it is not possible to obtain C by deleting A->C. This means we can not delete A->C
- **For AC->D :** We find $(AC)^+$ by assuming that we delete AC->D temporarily. We get $(AC)^+$={AC}. Thus by such deletion it is not possible to obtain D. This means we can not delete AC->D
- **For B->A :** We find $(B)^+$ by assuming that we delete B->A temporarily. We get $(B)^+$={BDE}. Thus by such deletion it is not possible to obtain A. This means we can not delete B->A
- **For B->D :** We find $(B)^+$ by assuming that we delete B->D temporarily. We get $(B)^+$={BEACD}. This shows clearly that even if we delete B->D we can obtain D. This means we can delete B->A. Thus it is redundant.
- **For B->E :** We find $(B)^+$ by assuming that we delete B->E temporarily. We get $(B)^+$={BDAC}. Thus by such deletion it is not possible to obtain E. This means we can not delete B->E

To summarize we get now

A->C
AC->D
B->A
B->E

Thus R.H.S gets simplified.

Step 3 : Now we will simplify L.H.S.

Consider AC->D. Here we can split A and C. For that we find closure set of A and C.

$(A)^+ = (AC)$

$(C)^+ = (C)$

Thus C can be obtained from both A as well as C. That also means we need not have to have AC on L.H.S. Instead, only A can be allowed and C can be eliminated. Thus after simplification we get

A->D

To summarize we get now

A->C
A->D
B->A
B->E

Thus L.H.S gets simplified.

Step 3 : The simplified L.H.S. and R.H.S can be combined together to form

A->CD
B->AE

This is a **minimal cover** or **Canonical cover** of functional dependencies.

4.3 Concept of Redundancy and Anomalies

Definition : Redundancy is a condition created in database in which same piece of data is held at two different places.

Redundancy is at the root of several problems associated with relational schemas.

Problems caused by redundancy : Following problems can be caused by redundancy-

i) **Redundant storage :** Some information is stored repeatedly.

ii) **Update anomalies :** If one copy of such repeated data is updated then inconsistency is created unless all other copies are similarly updated.

iii) **Insertion anomalies :** Due to insertion of new record repeated information get added to the relation schema.

iv) **Deletion anomalies :** Due to deletion of particular record some other important information associated with the deleted record get deleted and thus we may lose some other important information from the schema.

Example : Following example illustrates the above discussed anomalies or redundancy problems Consider following Schema in which all possible information about Employee is stored.

EmpID	EName	Salary	DeptID	DeptName	DeptLoc
1	AAA	10000	101	XYZ	Pune
2	BBB	20000	101	XYZ	Pune
3	CCC	30000	101	XYZ	Pune
4	DDD	40000	102	PQR	Mumbai

Redundancy!!!

1) **Redundant storage :** Note that the information about **DeptID, DeptName** and **DeptLoc** is repeated.
2) **Update anomalies :** In above table if we change **DeptLoc** of Pune to Chennai, then it will result **inconsistency** as for DeptID 101 the DeptLoc is Pune. Or otherwise, we need to **update multiple copies** of **DeptLoc** from Pune to Chennai. Hence this is an update anomaly.
3) **Insertion anomalies :** For above table if we want to add new tuple say (5, EEE, 50000) for **DeptID 101** then it will cause repeated information of (101, XYZ, Pune) will occur.
4) **Deletion anomalies :** For above table, if we delete a record for **EmpID 4,** then automatically information about the **DeptID 102, DeptName PQR** and **DeptLoc Mumbai** will get deleted and one may not be aware about **DeptID 102.** This causes deletion anomaly.

4.4 Decomposition

- Decomposition is the process of breaking down one table into multiple tables.
- **Formal definition of decomposition is -**
- A decomposition of relation Schema R consists of replacing the relation Schema by two relation schema that each contain a subset of attributes of R and together include all attributes of R by storing projections of the instance.
- For example - Consider the following table

Employee_Department table as follows -

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production

E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Hyderabad	25000	D005	Human Resource

We can decompose the above relation Schema into two relation schemas as **Employee (Eid, Ename, Age, City, Salary)** and **Department (Deptid, Eid, DeptName)** as follows -

Employee Table

Eid	Ename	Age	City	Salary
E001	ABC	29	Pune	20000
E002	PQR	30	Pune	30000
E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Hyderabad	25000

Department Table

Deptid	Eid	DeptName
D001	E001	Finance
D002	E002	Production
D003	E003	Sales
D004	E004	Marketing
D005	E005	Human Resource

- The decomposition is used for eliminating redundancy.
- **For example :** Consider following relation **Schema R** in which we assume that the grade determines the salary, the redundancy is caused

Schema R

Name	eid	deptname	Grade	Salary
AAA	121	Accounts	2	8000
AAA	132	Sales	3	7000
BBB	101	Marketing	4	7000
CCC	106	Purchase	2	8000

Redundancy!!!

- Hence, the above table can be decomposed into two Schema S and T as follows :

Schema S

Name	eid	deptname	Grade
AAA	121	Accounts	2
AAA	132	Sales	3
BBB	101	Marketing	4
CCC	106	Purchase	2

Schema T

Grade	Salary
2	8000
3	7000
4	7000
2	8000

Problems Related to Decomposition :

Following are the potential problems to consider :

1) Some queries become more **expensive**.
2) Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
3) Checking some dependencies may require joining the instances of the decomposed relations.
4) There may be loss of information during decomposition.

Properties Associated With Decomposition

There are two properties associated with decomposition and those are –

1) **Loss-less Join or non Loss Decomposition :** When all information found in the original database is preserved after decomposition, we call it as loss less or non loss decomposition.
2) **Dependency Preservation :** This is a property in which the constraints on the original table can be maintained by simply enforcing some constraints on each of

4.5 Loss-Less Join

GTU : Dec.-05, May -12, Marks 7

The lossless join can be defined using following three conditions :

i) **Union** of attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.

$Att(R1) \cup Att(R2) = Att(R)$

ii) **Intersection** of attributes of R1 and R2 must not be NULL.

$Att(R1) \cap Att(R2) \neq \Phi$

iii) **Common attribute** must be a key for at least one relation (R1 or R2)

$Att(R1) \cap Att(R2) \rightarrow Att(R1)$

or $Att(R1) \cap Att(R2) \rightarrow Att(R2)$

Example 4.5.1 *Consider the following relation R(A, B, C, D)and FDs A->BC, is the decomposition of R into R1(A, B, C), R2(A, D). Check if the decomposition is lossless join or not.*

Solution :

Step 1 : Here $Att(R1) \cup Att(R2) = Att(R)$ i.e $R1(A,B,C) \cup R2(A,D) = (A,B,C,D)$ i.e R.

Thus first condition gets satisfied.

Step 2 : Here $R1 \cap R2 = \{A\}$. Thus $Att(R1) \cap Att(R2) \neq \phi$. Here the second condition gets satisfied.

Step 3 : $Att(R1) \cap Att(R2) \rightarrow \{A\}$. Now $(A)^+ = \{A,B,C\}$ ϕ attributes of R1. Thus the third condition gets satisfied.

This shows that the given decomposition is a **lossless join**.

Example 4.5.2 *Consider the following relation R(A, B, C, D, E, F) and FDs A->BC, C->A, D->E, F->A, E->D is the decomposition of R into R1(A, C, D), R2(B, C, D), and R3(E, F, D). Check for lossless.*

Solution :

Step 1 : $R1 \cup R2 \cup R3 = R$. Here the first condition for checking lossless join is satisfied as $(A,C,D) \cup (B,C,D) \cup (E,F,D) = \{A,B,C,D,E,F\}$ which is nothing but R.

Step 2 : Consider $R1 \cap R2 = \{CD\}$ and $R2 \cap R3 = \{D\}$. Hence second condition of intersection not being Φ gets satisfied.

Step 3 : Now, consider R1(A, C, D) and R2(B, C, D). We find $R1 \cap R2 = \{CD\}$

$(CD)^+ = \{ABCDE\} \in$ attributes of R1 i.e.{A, C, D}. Hence condition 3 for checking lossless join for R1 and R2 gets satisfied.

Step 4 : Now, consider R2(B, C, D) and R3(E, F, D) . We find $R2 \cap R3 = \{D\}$.

$(D)^+ = \{D, E\}$ which is neither complete set of attributes of R2 or R3.

[Note that F is missing for being attribute of R3].

Hence it is not **lossless join decomposition**. Or in other words we can say it is a **lossy decomposition**.

Example 4.5.3 *Suppose that we decompose schema R = (A,B,C,D,E) into (A,B,C) (C,D,E) Show that it is not a lossless decomposition.*

Solution :

Step 1 : Here we need to assume some data for the attributes A, B, C, D, and E. Using this data we can represent the relation as follows –

Relation R

A	B	C	D	E
a	1	x	p	q
b	2	x	r	s

Relation R1 = (A,B,C)

A	B	C
a	1	x
b	2	x

Relation R2 = (C,D,E)

C	D	E
x	p	q
x	r	s

Step 2 : Now we will join these tables using natural join, i.e. the join based on common attribute C. We get R1 ⋈ R2 as

A	B	C	D	E
a	1	x	p	q
a	1	x	r	s
b	2	x	p	q
b	2	x	r	s

Here we get more rows or tuples than original relation R

Clearly R1 ⋈ R2 ⊄ R. Hence it is not lossless decomposition.

Review Questions

1. *Why do we decompose any relation ? When we decompose any relation, which properties need to be preserved ? Explain with example.* **GTU : Dec.-05, Marks 6**
2. *What is functional dependency ? Explain non-loss decomposition.* **GTU : May -12, Marks 7**

4.6 Dependency Preservation

- **Definition :** A Decomposition D = {R1, R2, R3....Rn} of R is dependency preserving for a set F of Functional dependency if - (F1 ∪ F2 ∪ ... ∪ Fm) = F.
- If decomposition is not dependency-preserving, some dependency is lost in the decomposition.

Example 4.6.1 *Consider the relation R (A, B, C) for functional dependency set {A -> B and B ->C} which is decomposed into two relations R1 = (A, C) and R2 = (B, C). Then check if this decomposition dependency preserving or not.*

Solution : This can be solved in following steps :

Step 1 : For checking whether the decomposition is dependency preserving or not we need to check following condition

$$F^+ = (F1 \cup F2)^+$$

Step 2 : We have with us the F^+ ={ A->B and B->C }

Step 3 : Let us find $(F1)^+$ for relation R1 and $(F2)^+$ for relation R2

R1(A,C)	
A->A	Trivial
C->C	Trivial
A->C	∵ In $(F)^+$ A->B->C and it is Nontrivial
AC->AC	Trivial
A->B	but is not useful as B is not part of R1 set
We can not obtain C->A	

R2(B,C)	
B->B	Trivial
C->C	Trivial
B->C	∵ In $(F)^+$ B->C and it is Non-Trivial
BC->BC	Trivial
We can not obtain C->B	

Step 4 : We will eliminate all the trivial relations and useless relations. Hence we can obtain R1 and R2 as

R1(A,C)	
A->C	Nontrivial

R2(B,C)	
B->C	Non-Trivial

$(F1 \cup F2)^+ = \{A->C, B->C\} \neq \{A->B, B->C\}$ i.e. $(F)^+$

Thus the condition specified in step 1 i.e. $F^+ = (F1 \cup F2)^+$ **is not true.** Hence it is **not dependency preserving decomposition.**

Example 4.6.2 *Let relation R(A, B, C, D) be a relational schema with following functional dependencies {A->B, B->C, C->D, and D->B}. The decomposition of R into (A, B), (B, C) and (B, D). Check whether this decomposition is dependency preserving or not.*

Solution :

Step 1 : Let $(F)^+ = \{A->B, B->C, C->D, D->B\}$.

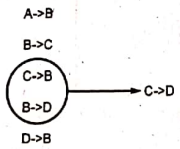
Step 2 : We will find $(F1)^+$, $(F2)^+$, $(F3)^+$ for relations R1(A,B), R2(B,C) and R3(B,D) as follows.

R1(A,B)	R2(B,C)	R3(B,D)
A->A Trivial	B->B Trivial	B->B Trivial
B->B Trivial	C->C Trivial	D->D Trivial
A->B ∵ $(F)^+$ and it's non Trivial	B->C ∵ $(F)^+$ and it's non Trivial	B-> D ∵ $(F)^+$ as and B->C->D and it's non Trivial
B->A can not be obtained	C->B ∵ In $(F)^+$ and C->D->C and it is Nontrivial	D->B ∵ $(F)^+$ and it's non Trivial
AB->AB	BC->BC Trivial	BD->BD Trivial

Step 3 : We will eliminate all the trivial relations and useless relations. Hence we can obtain R1 ∪ R2 ∪ R3 as

R1(A,B)	R2(B,C)	R2(B,D)
A->B	B->C	B-> D
	C->B	D->B

Step 4 : As from above FD's we get



Step 5 : This proves that $F^+ = (F1 \cup F2 \cup F3)^+$. Hence given **decomposition is dependency preserving.**

4.7 Normal Forms

GTU : Dec.-05, Summer-13, 14, 17, Winter-13, 15, 17, June-05, May-11, Marks 8

- Normalization is the process of reorganizing data in a database so that it meets **two basic requirements :**
  1) There is **no redundancy** of data (all data is stored in only one place), and
  2) **Data dependencies** are logical (all related data items are stored together)

Need for Normalization

(1) **It eliminates redundant data**
(2) **It reduces chances of data error**
(3) The normalization is important because it allows database to take up **less disk space.**
(4) It also help in increasing the **performance.**
(5) It improves the data integrity and consistency.

4.7.1 First Normal Form

The table is said to be in 1NF if it follows following rules -

i) It should only have single (atomic) valued attributes/columns.
ii) Values stored in a column should be of the same domain
iii) All the columns in a table should have unique names.
iv) And the order in which data is stored, does not matter.

Consider following Student table

Student

sid	sname	Phone
1	AAA	11111 22222
2	BBB	33333
3	CCC	44444 55555

As there are multiple values of phone number for sid 1 and 3, the above table is not in 1NF. We can make it in 1NF. The conversion is as follows –

sid	sname	Phone
1	AAA	11111
1	AAA	22222
2	BBB	33333
3	CCC	44444
3	CCC	55555

4.7.2 Second Normal Form

Before understanding the second normal form let us first discuss the concept of partial functional dependency and prime and non prime attributes.

Concept of Partial Functional Dependency

Partial dependency means that a nonprime attribute is functionally dependent on part of a candidate key.

For example : Consider a relation R(A,B,C,D) with functional dependency

{AB->CD, A->C}

Here (AB) is a candidate key because

$(AB)^+ = \{ABCD\} = \{R\}$

Hence {A,B} are prime attributes and {C,D} are non prime attribute. In A->C, the non prime attribute C is dependent upon A which is actually a part of candidate key AB. Hence due to A->C we get partial functional dependency.

Prime and Non Prime Attributes

- **Prime attribute :** An attribute, which is a part of the candidate-key, is known as a prime attribute.
- **Non-prime attribute :** An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.
- **Example : Consider a Relation** R = {A,B,C,D} and candidate key as AB, the Prime attributes : A, B

  Non Prime attributes : C, D

The Second Normal Form

For a table to be in the Second Normal Form, following conditions must be followed

i) It should be in the First Normal form.

ii) It should not have partial functional dependency.

For example : Consider following table in which every information about a the Student is maintained in a table such as student id(sid), student name(sname), course id(cid) and course name(cname).

Student_Course

sid	sname	cid	cname
1	AAA	101	C
2	BBB	102	C++
3	CCC	101	C
4	DDD	103	Java

This table is not in 2NF. For converting above table to 2NF we must follow the following steps -

Step 1 : The above table is in 1NF.

Step 2 : Here **sname** and **sid** are associated similarly **cid** and **cname** are associated with each other. Now if we delete a record with **sid = 2**, then automatically the course C++ will also get deleted. Thus,

sid->sname or **cid->cname** is a partial functional dependency, because **{sid,cid}** should be essentially a candidate key for above table. Hence to bring the above table to 2NF we must decompose it as follows :

Student

sid	sname	cid

Here candidate key is (sid, cid) and (sid, cid)- > sname

1	AAA	101
2	BBB	102
3	CCC	101
4	DDD	103

Course

cid	cname
101	C
102	C++
101	C
103	Java

Here candidate key is cid here cid->cname

Thus now table is in 2NF as there is no partial functional dependency.

Example 4.7.1 *Study the relation given below and state what level of normalization can be achieved and normalize it upto that level.*

Order no.	Order Date	Item Lines		
		Item Code	Quantity	Price/Unit
1456	26-12-1999	3687	52	50.4
		4627	38	60
		3214	20	20.00
1886	04-03-1999	4629	45	20.25
		4627	30	60.20
1788	04-04-1999	4627	40	60.20

GTU : Dec.-05, Marks 3

Solution :

Reason for the given relation being unnormalized

1. Observe order for many items
2. Item lines has many attributes-called composite attributes
3. Each tuple has variable length
4. Difficult to store due to non-uniformity
5. Given item code difficult to find qty-ordered and hence called Unnormalized relation

For conversion to First Normal Form –

- Identify the composite attributes, convert the composite attributes to individual attributes.
- Duplicate the common attributes as many times as lines in composite attribute.
- Every attribute now describes single property and not multiple properties, some data will be duplicated.
- Now this is called First normal form (1NF) also called flat file.

Order no.	Order Date	Item Lines		
		Item Code	Quantity	Price/Unit
1456	26-12-1999	3687	52	50.4
1456	26-12-1999	4627	38	60
1456	26-12-1999	3214	20	20.00
1886	04-03-1999	4629	45	20.25
1886	04-03-1999	4627	30	60.20
1788	04-04-1999	4627	40	60.20

Fig. 4.7.1 Table in First Normal Form

- The above table has insertion, deletion and update anomalies. For instance – if we delete order no. 1886, then the item code 4629 gets lost. Similarly if we update 4627, then all instances of 4627 need to be changed.
- We need to convert 2NF if it is in 1NF. The non-key attributes are functionally dependent on key attribute and if there is a composite key then no non-key attribute is functionally depend on one part of the key.
- The table can be converted to 2NF as follows –

Orders

OrderNo	OrderDate
1456	26-12-1999
1886	04-03-1999
1788	04-04-1999

Order Details

OrderNo	ItemCode	Qty
1456	3687	52
1886	4629	45
1788	4627	40

Prices

ItemCode	Price/Unit
3687	50.4
4627	60
3214	20
4629	20.25

4.7.3 Third Normal Form

Before understanding the third normal form let us first discuss the concept of transitive dependency, super key and candidate key

Concept of Transitive Dependency

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. For example -

X -> Z is a transitive dependency if the following functional dependencies hold true :

X->Y

Y->Z

Concept of Super key and Candidate Key

Superkey : A super key is a set or one of more columns (attributes) to uniquely identify rows in a table.

Candidate key : The minimal set of attribute which can uniquely identify a tuple is known as candidate key. For example consider following table

RegID	RollNo	Sname
101	1	AAA
102	2	BBB
103	3	CCC
104	4	DDD

Superkeys

- {RegID}
- {RegID, RollNo}
- {RegID, Sname}
- {RollNo, Sname}
- {RegID, RollNo, Sname}

Candidate Keys

- {RegID}
- {RollNo}

Third Normal Form

A table is said to be in the Third Normal Form when,

i) It is in the Second Normal form.(i.e. it does not have partial functional dependency)

ii) It doesn't have transitive dependency.

Or in other words

In other words 3NF can be defined as : A table is in 3NF if it is in 2NF and for each functional dependency

X-> Y

at least one of the following conditions hold :

i) X is a super key of table

ii) Y is a prime attribute of table

For example : Consider following table **Student_details** as follows -

sid	sname	zipcode	cityname	state
1	AAA	11111	Pune	Maharashtra
2	BBB	22222	Surat	Gujarat
3	CCC	33333	Chennai	Tamilnadu
4	DDD	44444	Jaipur	Rajastan
5	EEE	55555	Mumbai	Maharashtra

Here

Super keys : {sid},{sid,sname},{sid,sname,zipcode}, {sid,zipcode,cityname}... and so on.

Candidate keys : {sid}

Non-Prime attributes : {sname,zipcode,cityname,state}

The dependencies can be denoted as

sid->sname

sid->zipcode

zipcode->cityname

cityname->state

The above denotes the transitive dependency. Hence above table is not in 3NF. We can convert it into 3NF as follows :

Student

sid	sname	zipcode
1	AAA	11111
2	BBB	22222
3	CCC	33333
4	DDD	44444
5	EEE	55555

Zip

zipcode	cityname	state
11111	Pune	Maharashtra
22222	Surat	Gujarat
33333	Chennai	Tamilnadu
44444	Jaipur	Rajasthan
55555	Mumbai	Maharashtra

Example 4.7.2 *Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F = {{A, B} → C, A→ {D, E}, B→ F, F→{G, H}, D→{I, J} }*

1. What is the key for R ? Demonstrate it using the inference rules.

2. Decompose R into 2NF, then 3NF relations.

Solution : Let,

$A \rightarrow DE$ (given)

$\therefore$ $A \rightarrow D$, $A \rightarrow E$ (decomposition rule)

As $D \rightarrow IJ$, $A \rightarrow IJ$

Using union rule we get

$A \rightarrow DEIJ$

As $A \rightarrow A$

we get $A \rightarrow ADEIJ$

Using augmentation rule we compute AB

$AB \rightarrow ABDEIJ$

But $AB \rightarrow C$ (given)

$\therefore$ $AB \rightarrow ABCDEIJ$

$B \rightarrow F$ (given) $F \rightarrow GH$ $\therefore$ $B \rightarrow GH$ (transitivity)

$\therefore$ $AB \rightarrow AGH$ is also true

Similarly $AB \rightarrow AF$ $\because B \rightarrow F$ (given)

Thus now using union rule

$AB \rightarrow ABCDEFGHIJ$

$\therefore$ AB is a key

The table can be converted to 2NF as

$R_1 = (\underline{A}, \underline{B}, C)$

$R_2 = (\underline{A}, D, E, I, J)$

$R_3 = (\underline{B}, F, G, H)$

The above 2NF relations can be converted to 3NF as follows

$R_1 = (\underline{A}, \underline{B}, C)$

$R_2 = (\underline{A}, D, E)$

$R_3 = (\underline{D}, I, J)$

$R_4 = (\underline{B}, E)$

$R_5 = (E, G, H)$.

Example 4.7.3 *A software contract and consultancy firm maintains details of all the various projects in which its employees are currently involved. These details comprise :*

- *Employee Number*
- *Employee Name*
- *Date of Birth*
- *Department Code*
- *Department Name*
- *Project Code*
- *Project Description*
- *Project Supervisor*

Assume the following :

- *Each employee number is unique.*
- *Each department has a single department code.*
- *Each project has a single code and supervisor.*
- *Each employee may work on one or more projects.*
- *Employee names need not necessarily be unique.*
- *Project Code, Project Description and Project Supervisor are repeating fields.*

Normalise this data to Third Normal Form. **GTU : Summer-17. Marks 4**

Solution :

Un-Normalized Form

Employee Number, Employee Name, Date of Birth, Department Code, Department Name, Project Code, Project Description, Project Supervisor

1NF

Employee Number, Employee Name, Date of Birth
Department Code, Department Name
Employee Number, Project Code, Project Description, Project Supervisor

2NF

Employee Number, Employee Name, Date of Birth, Department Code, Department Name
Employee Number, Project Code,
Project Code, Project Description, Project Supervisor

3NF

Employee Number, Employee Name, Date of Birth, *Department Code
Department Code, Department Name
Employee Number, Project Code
Project Code, Project Description, Project Supervisor

Example 4.7.4 *What is normalization ?Normalize below given relation upto 3NF*
STUDENT.

StudID	StudName	City	Pincode	ProjectID	ProjectName	Course	Content
S101	Ajay	Surat	326201	P101	Health	Programming	C++, Java, C
S102	Vijay	Pune	325456	P102	Social	WEB	HTML, PHP, ASP

GTU : Winter-17, Marks 7

Solution : For converting the given schema to first normal form, we will arrange it in such a way that have each tuple contains single record. For that purpose we need to split the schema into two tables namely **Student** and **Projects**

1NF

Student

StudID	StudName	Pincode	City
S101	Ajay	326201	Surat
S102	Vijay	325456	Pune

Projects

StudID	ProjectID	ProjName	Course	Content
S101	P101	Health	Programming	C++
S101	P101	Health	Programming	Java
S101	P101	Health	Programming	C
S102	P102	Social	WEB	HTML
S102	P102	Social	WEB	PHP
S102	P102	Social	WEB	ASP

2NF

For a table to be in 2NF, there should not be any partial dependency.

Student

StudID	StudName	Pincode	City
S101	Ajay	326201	Surat
S102	Vijay	325456	Pune

Project

StudID	ProjectID	ProjName	CourseID
S101	P101	Health	C101
S101	P101	Health	C102
S101	P101	Health	C103
S102	P102	Social	C104
S102	P102	Social	C105
S102	P102	Social	C106

CourseDetails

CourseID	Course	Content
C101	Programming	C++
C102	Programming	Java

C103	Programming	C
C104	WEB	HTML
C105	WEB	PHP
C106	WEB	ASP

3NF

There was a transitive dependency in 2NF tables because city is associated with student ID and city depends upon zip code. Hence the transitive dependency is removed to covert table into 3NF. The required 3NF schema is as below -

Student

StudID	StudName	Pincode
S101	Ajay	326201
S102	Vijay	325456

Student_Address

Pincode	City
326201	Surat
325456	Pune

Project

StudID	ProjectID	ProjName	CourseID
S101	P101	Health	C101
S101	P101	Health	C102
S101	P101	Health	C103
S102	P102	Social	C104
S102	P102	Social	C105
S102	P102	Social	C106

CourseDetails

CourseID	Course	Content
C101	Programming	C++
C102	Programming	Java
C103	Programming	C
C104	WEB	HTML
C105	WEB	PHP
C106	WEB	ASP

4.7.4 Boyce / Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a **higher version** of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF.

A 3NF table which **does not have multiple overlapping** candidate keys is said to be in BCNF.

Or in other words,

For a table to be in BCNF, following conditions must be satisfied :

i) R must be in 3rd Normal Form

ii) For each functional dependency ($X \rightarrow Y$), X should be a super Key. In simple words if Y is a prime attribute then X can not be non prime attribute.

For example - Consider following table that represents that a Student enrollment for the course -

Enrollment Table

sid	Course	Teacher
1	C	Ankita
1	Java	Poonam
2	C	Ankita
3	C++	Supriya
4	C	Archana

From above table following observations can be made :

- One student can enroll for multiple courses. For example student with sid=1 can enroll for C as well as Java.
- For each course, a teacher is assigned to the student.
- There can be multiple teachers teaching one course for example course C can be taught by both the teachers namely - Ankita and Archana.
- The candidate key for above table can be (sid, course), because using these two columns we can find
- The above table holds following dependencies
  - (sid, course)->Teacher
  - Teacher->course
- The above table is not in BCNF because of the dependency teacher->course. Note that the teacher is not a superkey or in other words, teacher is a non prime attribute and course is a prime attribute and non-prime attribute derives the prime attribute.
- To convert the above table to BCNF we must decompose above table into Student and Course tables

Student

sid	Teacher
1	Ankita
1	Poonam
2	Ankita
3	Supriya
4	Archana

Course

Teacher	course
Ankita	C
Poonam	Java
Ankita	C
Supriya	C++
Archana	C

Now the table is in BCNF

Example 4.7.5 *Consider a relation(A,B,C,D) having following FDs.{AB->C, AB->D, C->A, B->D}. Find out the normal form of R.*

Solution :

Step 1 : We will first find out the candidate key from the given FD.

$(AB)^+ = \{ABCD\} = R$

$(BC)^+ = \{ABCD\} = R$

$(AC)^+ = \{AC\} \neq R$

There is no involvement of D on LHS of the FD rules. Hence D can not be part of any candidate key. Thus we obtain two candidate keys $(AB)^+$ and $(BC)^+$. Hence

prime attributes = {A,B,C}

Non prime attributes = {D}

Step 2 : Now, we will start checking from reverse manner, that means from BCNF, then 3NF, then 2NF.

Step 3 : For R being in BCNF for X->Y the X should be candidate key or super key.

From above FDs consider C->D in which C is not a candidate key or super key. Hence given relation is not in BCNF.

Step 4 : For R being in 3NF for X->Y either i) the X should be candidate key or super key or ii) Y should be prime attribute. (For prime and non prime attributes refer step 1)

- For AB->C or AB->D the AB is a candidate key. Condition for 3NF is satisfied.
- Consider C->A. In this FD the C is not candidate key but A is a prime attribute. Condition for 3NF is satisfied.
- Now consider B->D. In this FD, the B is not candidate key, similarly D is not a prime attribute. Hence condition for 3NF fails over here.

Hence given relation is not in 3NF.

Step 5 : For R being in 2NF following condition **should not occur.**

Let X->Y, if X is a proper subset of candidate key and Y is a non prime attribute. This is a case of partial functional dependency.

For relation to be in 2NF there should not be any partial functional dependency.

- For AB->C or AB->D the AB is a **complete candidate key**. Condition for 2NF is satisfied.
- Consider C->A. In this FD the C is not candidate key. Condition for 2NF is satisfied.
- Now consider B->D. In this FD, the B is a part of candidate key(AB or BC), similarly D is not a prime attribute. That means partial functional dependency occurs here.

Hence condition for 2NF fails over here.

Hence given relation is not in 2NF.

Therefore we can conclude that the given relation R **is in 1NF.**

Example 4.7.6 *Consider a relation R(ABC) with following FD A->B, B->C and C->A. What is the normal form of R ?*

Solution :

Step 1 : We will find the candidate key

$(A)^+ = \{ABC\} = R$

$(B)^+ = \{ABC\} = R$

$(C)^+ = \{ABC\} = R$

Hence A, B and C all are candidate keys

Prime attributes = {A,B,C}

Non prime attribute{}

Step 2 : For R being in BCNF for X->Y the X should be candidate key or super key.

From above FDs

- Consider A->B in which A is a candidate key or super key. Condition for BCNF is satisfied.
- Consider B->C in which B is a candidate key or super key. Condition for BCNF is satisfied.
- Consider C->A in which C is a candidate key or super key. Condition for BCNF is satisfied.

This shows that the given relation R **is in BCNF.**

Example 4.7.7 *Consider table R(A,B,C,D,E) with FDs as A->B, BC->E, and ED->A.The table is in which normal form ? Justify your answer.* **GTU : Summer-13, Marks 7**

Solution :

Step 1 : We will first find out the candidate keys for given relation R

$(ACD)^+ = \{A,B,C,D,E\}$

$(BCD)^+ = \{A,B,C,D,E\}$

$(CDE)^+ = \{A,B,C,D,E\}$

Step 2 : Let A->B, the ACD is candidate key and A is a partial key, B is a prime attribute(i.e. it is also part of candidate key). Hence A->B is not a partial functional dependency.

Similarly in BC->E and ED->A,

E and A are prime-attributes and hence both are not partial functional dependencies.

Hence R is in 2NF.

Step 3 : According to 3NF, every non-prime attribute must be dependent on the candidate key.

In the given functional dependencies, all dependent attributes are prime-attributes. Hence the relation R is in 3NF.

Step 4 : For R being in BCNF for X->Y the X should be candidate key or super key.

The table is not in BCNF, none of A, BC and ED contain a key.

Example 4.7.8 *A college maintains details of its lecturers' subject area skills. These details comprise :*

- *Lecturer Number*
- *Lecturer Name*
- *Lecturer Grade*
- *Department Code*
- *Department Name*
- *Subject Code*
- *Subject Name*
- *Subject Level*

Assume that each lecturer may teach many subjects but may not belong to more than one department.

Subject Code, Subject Name and Subject Level are repeating fields.

Normalise this data to Third Normal Form. **GTU : Summer-17, Marks 4**

Solution :

Unnormalized Form

Lecturer Number, Lecturer Name, Lecturer Grade, Department Code, Department Name, Subject Code, Subject Name, Subject Level

1NF

LecturerNumber	Lecturer Name	Lecturer Grade	Department Code	Department Name

Lecturer Number	Subject Code	Subject Name	Subject Level

2NF

Lecturer Number	_Lecturer Name	_Lecturer Grade	_Department Code	_Department Name

Lecturer Number	Subject Code

Subject Code	_Subject Name	_Subject Level

3NF

Lecturer Number	Lecturer Name	Lecturer Grade

*Department Code

Department Code	_Department Name

Lecturer Number	Subject Code

Subject Code	Subject Name	Subject Level

Example 4.7.9 *Prove that any relational schema with two attributes is in BCNF.*

Solution : Here, we will consider R={A,B} i.e. a relational schema with two attributes. Now various possible FDs are A->B, B->A.

From the above FDs

- Consider A->B in which A is a candidate key or super key. Condition for BCNF is satisfied.
- Consider B->A in which B is a candidate key or super key. Condition for BCNF is satisfied.
- Consider both A->B and B->A with both A and B is candidate key or super key. Condition for BCNF is satisfied.
- No FD holds in relation R. In this {A,B} is candidate key or super key. Still condition for BCNF is satisfied.

This shows that any relation R **is in BCNF** with two attributes.

Example 4.7.10 *Prove the statement "Every relation which is in BCNF is in 3NF but the converse is not true".* **GTU : Winter-15, Marks 4**

Solution : For a relations to be in 3NF

A table is said to be in the Third Normal Form when,

i) It is in the Second Normal form. (i.e. it does not have partial functional dependency)

ii) It doesn't have transitive dependency.

Or in other words

In other words 3NF can be defined as : A table is in 3NF if it is in 2NF and for each functional dependency

X-> Y

at least one of the following conditions hold :

iii) X is a super key of table

iv) Y is a prime attribute of table

For a relation to be in BCNF

(1) It should be in 3NF

(2) **A 3NF** table which **does not have multiple overlapping** candidate keys is said to be in BCNF.

For proving that the table can be in 3NF but not in BCNF consider Following relation

R(Student, Subject, Teacher) . Consider following are FDs

(Subject, Student)-> Teacher

Because subject and student combination gives unique teacher.

Teacher -> Subject

Because each teacher teaches only Subject.

(Teacher, Student)->Subject

- So, this relation is in 3NF as every non-key attribute is non-transitively fully functional dependent on the primary key.
- But it is not in BCNF. Because this is a case of **overlapping of candidate keys** because there are two composite candidate keys :
  - (Subject, Student)
  - (Teacher, Student)

  And Student is a common attribute in both the candidate keys.

So we need to normalize the above table to BCNF. For that purpose we must set Teacher to be a candidate key

The decomposition of above takes place as follows

R1(Student, Teacher)

R2(Teacher, Subject)

Now table is in 3NF, as well as in BCNF.

This show that the relation Every relation which is in BCNF is in 3NF but the converse is not true

4.7.5 Multivalued Dependencies and Fourth Normal Form

Concept of Multivalued Dependencies

- A table is said to have multi-valued dependency, if the following conditions are true,

1) For a dependency A → B, if for a single value of A, **multiple values** of B exists, then the table may have multi-values dependency.
2) Also, a table should have **at-least 3 columns** for it to have a multi-valued dependency.
3) And, for a relation R(A,B,C), if there is a multi-valued **dependency between,** A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

- In simple terms, if there are two columns A and B - and for column A if there are multiple values of column B then we say that MVD exists between A and B
- The multivalued dependency is denoted by ↠
- If there exists a multivalued dependency then the table is **not in 4th normal form.**
- **For example :** Consider following table for information about student

Student

sid	Course	Skill
1	C	English
	C++	German
2	Java	English
		French

Here sid = 1 leads to multiple values for courses and skill. Following table shows this

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Here **sid** and **course** are dependent but the **Course** and **Skill** are independent. The multivalued dependency is denoted as :

sid ↠ Course

sid ↠ Skill

Fourth Normal Form

Definition : For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions :

1) It should be in the Boyce-Codd Normal Form(BCNF).
2) And, the table should not have any multi-valued dependency.

For example : Consider following student relation which is not in 4NF as it contains multivalued dependency.

Student Table

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Now to convert the above table to 4NF we must decompose the table into following two tables.

Student_Course Table

Key : (sid,Course)

sid	Course
1	C
1	C++
2	Java

Student_Skill Table

Key : (sid,Skill)

sid	Skill
1	English
1	German
2	English
2	French

Thus the tables are now in 4NF.

Review Questions

1. *Define 2NF. Taking suitable example explain update anomalies in it* **GTU : June-05, Marks 8**
2. *Explain MVD and 4NF* **GTU : June-05, Marks 8**
3. *Explain BCNF with example. For a relation to be in BCNF is it required to be in 3NF ?* **GTU : Dec.-05, Marks 6**
4. *Explain 2NF with example.* **GTU : Summer-13, Marks 3**
5. *Explain BCNF with example.* **GTU : May-11,Summer-13, Marks 3**
6. *What is an anomaly in database design ? How it can be solved? Explain BCNF with suitable example* **GTU : Winter-13, Marks 7**
7. *What is normalization ? Why normalization process is needed ? Explain 1NF, 2NF, 3NF with example* **GTU : Summer-14, Marks 7**

4.8 Concept of Join Dependencies **GTU : May-12, Marks 7**

- Join decomposition is a further generalization of Multivalued dependencies.
- If the join of R1 and R2 over C is equal to relation R, then we can say that a Join Dependency (JD) exists.
- Where R1 and R2 are the decompositions R1(A, B, C) and R2(C, D) of a given relations R (A, B, C, D).
- Alternatively, R1 and R2 are a lossless decomposition of R.
- A JD ⋈ {R1, R2,..., Rn} is said to hold over a relation R if R1, R2,....., Rn is a lossless-join decomposition.
- The *(A, B, C, D), (C, D) will be a JD of R if the join of join's attribute is equal to the relation R.
- Here, *(R1, R2, R3) is used to indicate that relation R1, R2, R3 and so on are a JD of R.

Concept of Fifth Normal Form

The database is said to be in 5NF if -

i) It is in 4th Normal Form

ii) If we can decompose table further to eliminate redundancy and anomalies and when we rejoin the table we should not be losing the original data or get a new record **(join Dependency Principle)**

The fifth normal form is also called as **project join normal form**

For example - Consider following table

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Britania	Biscuits

Here we assume the keys as{Seller, Company, Product}

The above table has multivalued dependency as

Seller↠{Company, Product}. Hence table is not in 4th Normal Form. To make the above table in 4th normal form we decompose above table into two tables as

Seller_Company

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Seller_Product

Seller	Product
Rupali	Cinthol
Sharda	Honey
Sharda	HairOil
Sharda	RoseWater
Sunil	Icecream
Sunil	Biscuits

The above table is in 4th Normal Form as there is no multivalued dependency. But it is not in 5th normal form because if we join the above two table we may get

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Amul	Biscuits
Sunil	Britania	Icecream
Sunil	Britania	Biscuits

Newly added records which are not present in original table

To avoid the above problem we can decompose the tables into three tables as

Seller_Company, Seller_Product, and Company Product table

Seller_Company

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Seller_Product

Seller	Product
Rupali	Cinthol
Sharda	Honey
Sharda	HairOil
Sharda	RoseWater
Sunil	Icecream
Sunil	Biscuit

Company_Product

Company	Product
Godrej	Cinthol
Dabur	Honey
Dabur	HairOil
Dabur	RoseWater
Amul	Icecream
Britania	Biscuit

Thus the table in in 5th normal form.

Review Question

1. *Why do we normalization ? Explain 4NF and 5NF.*

GTU : May-12, Marks 7

4.9 Oral Questions and Answers

Q.1 Define functional dependency.

Ans. : Let P and Q be sets of columns, then : P functionally determines Q, written P → Q if and only if any two rows that are equal on (all the attributes in) P must be equal on (all the attributes in) Q.

In other words, the functional dependency holds if

T1.P = T2.P, then T1.Q=T2.Q

Where notation T1.P projects the tuple T1 onto the attribute in P.

Q.2 Why certain functional dependencies are called trivial functional dependencies ?

Ans. : A functional dependency FD : X → Y is called trivial if Y is a subset of X. This kind of dependency is called trivial because it can be derived from common sense. If one "side" is a subset of the other, it's considered trivial. The left side is considered the determinant and the right the dependent.

For example - {A,B} -> B is a trivial functional dependency because B is a subset of A,B. Since {A,B} -> B includes B, the value of B can be determined. It's a trivial functional dependency because determining B is satisfied by its relationship to A,B.

Q.3 What is Armstrong Axiom ?

Ans. : The closure set of functional dependency can be computed using basic three rules which are also called as Armstrong's Axioms.

These are as follows -

i) **Reflexivity :** If X ⊇ Y, then X → Y

ii) **Augmentation :** If X → Y, then XZ → YZ for any Z

iii) **Transitivity :** If X → Y and Y → Z, then X → Z

In addition to above axioms some additional rules for computing closure set of functional dependency are as follows -

- **Union :** If X → Y and X → Z then X → YZ
- **Decomposition :** If X → YZ, then X → Y and X → Z

Q.4 What is redundancy ?

Ans. : Redundancy is a condition created in database in which same piece of data is held at two different places.

Redundancy is at the root of several problems associated with relational schemas.

Q.5 What are problems caused by redundancy ?

Ans. : Following problems can be caused by redundancy -

i) **Redundant storage :** Some information is stored repeatedly.

ii) **Update anomalies :** If one copy of such repeated data is updated then inconsistency is created unless all other copies are similarly updated.

iii) **Insertion anomalies :** Due to insertion of new record repeated information get added to the relation schema.

iv) **Deletion anomalies :** Due to deletion of particular record some other important information associated with the deleted record get deleted and thus we may lose some other important information from the schema.

Q.6 What is decomposition ?

Ans. : A decomposition of relation Schema R consists of replacing the relation Schema by two relation schema that each contain a subset of attributes of R and together include all attributes of R by storing projections of the instance.

Q.7 Name two properties associated with decomposition.

Ans. : (1) Lossless join and (2) Dependency preservation

Q.8 Define normalization.

Ans. : Normalization is the process of reorganizing data in a database so that it meets **two basic requirements :**

1) There is **no redundancy** of data (all data is stored in only one place), and

2) **data dependencies** are logical (all related data items are stored together)

Q.9 State anomalies of 1NF.

Ans. : All the insertion, deletion and update anomalies are in 1NF relation.

Q.10 Why 4NF in normal form is more desirable than BCNF ?

Ans. : 4NF is more desirable than BCNF because it reduces the repetition of information.

If we consider a BCNF schema not in 4NF we observe that decomposition into 4NF does not lose information provided that a lossless join decomposition is used, yet redundancy is reduced.

Q.11 Give an example of a relation schema R and set of dependencies such that R is in BCNF but not in 4NF.

Ans. : Consider relation R(A,B,C,D) with dependencies

$AB \rightarrow C$

$ABC \rightarrow D$

$AC \twoheadrightarrow B$

Here the only key is AB. Thus each functional dependency has superkey on the left. But MVD has non-superky on its left. So it is not 4NF.

Q.12 Show that if a relation is in BCNF, then it is also in 3NF.

Ans. :

- Boyce and Codd Normal Form is a **higher version** of the Third Normal form.
- **A 3NF** table which **does not have multiple overlapping** candidate keys is said to be in BCNF. When the table is in BCNF then it doesn't have partial functional dependency as well as transitive dependency.
- Hence it is true that if relation is in BCNF then it is also in 3NF.

Q.13 Why it is necessary to decompose a relation ?

Ans. :

- Decomposition is the process of breaking down one table into multiple tables.
- The decomposition is used for eliminating redundancy.

Q.14 Explain atleast two desirable properties of decomposition.

Ans. : There are two properties associated with decomposition and those are –

1) **Loss-less Join or non Loss Decomposition :** When all information found in the original database is preserved after decomposition, we call it as loss less or non loss decomposition.

2) **Dependency Preservation :** This is a property in which the constraints on the original table can be maintained by simply enforcing some constraints on each of the smaller relations.

Q.15 What is multivalued dependency ?

Ans. : A table is said to have multi-valued dependency, if the following conditions are true,

1) For a dependency $A \rightarrow B$, if for a single value of A, multiple values of B exists, then the table may have multi-values dependency.

2) Also, a table should have at-least 3 columns for it to have a multi-valued dependency.

3) And, for a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

❑❑❑

Notes

5 Query Processing and Optimization

Syllabus

Evaluation of relational algebra expressions, Query equivalence, Join strategies, Query optimization.

Contents

5.1 Basics of Query Processing

GTU : Dec.-05, 06, Mar.-10, May-12, June-08, Summer-14, 18, Winter-17,18, Marks 7

- Query processing is a collection of activities that are involved in extracting data from database.
- During query processing there is translation high level database language queries into the expressions that can be used at the physical level of filesystem.
- There are three basic steps involved in query processing and those are -

1. Parsing and translation :

- In this step the query is translated into its internal form and then into **relational algebra**.
- Parser checks syntax and verifies relations.
- For instance - If we submit the query as,

```
SELECT RollNo, name
FROM Student
HAVING RollNo=10
```

Then it will issue a syntactical error message as the correct query should be

```
SELECT RollNo, name
FROM Student
HAVING RollNo=10
```

Thus during this step the syntax of the query is checked so that only correct and verified query can be submitted for further processing.

2. Optimization :

- During this process the query evaluation plan is prepared from all the relational algebraic expressions.
- The query cost for all the evaluation plans is calculated.
- Amongst all equivalent evaluation plans the one with lowest cost is chosen.
- Cost is estimated using statistical information from the database catalog, such as the number of tuples in each relation, size of tuples, etc.

3. Evaluation :

- The query-execution engine takes a query-evaluation plan, executes that plan and returns the answers to the query.

The above describe steps are represented by following Fig. 5.1.1.

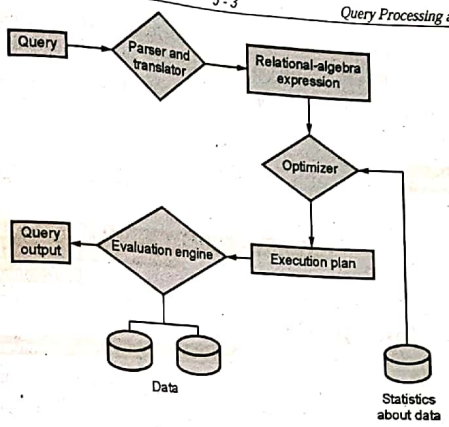


Fig. 5.1.1 Query processing

For example - If the SQL query is,

```
SELECT balance
FROM account
WHERE balance<1000
```

Step 1 : This query is first verified by the **parser and translator** unit for correct syntax. If so then the relational algebra expressions can be obtained. For the above given queries there are two possible relational algebra

(1) $\sigma_{balance<1000}(\pi_{balance}(account))$

(2) $\pi_{balance}(\sigma_{balance<1000}(account))$

Step 2 : Query evaluation plan : To specify fully how to evaluate a query, we need not only to provide the relational-algebra expression, but also to annotate it with instructions specifying how to evaluate each operation. For that purpose, using the order of evaluation of queries, two query evaluation plans are prepared. These are as follows

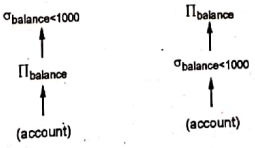


Fig. 5.1.2 Query evaluation plans

Associated with each query evaluation plan there is a **query cost**. The query optimization selects the query evaluation plan having minimum query cost.

Once the query plan is chosen, the **query is evaluated** with that plan and the **result of the query is output.**

Review Questions

1. *Explain steps of query processing with the help of neat diagram.* **GTU : Dec-05, 06 Marks 6, Mar.-10, Marks 7, May-12, Marks 4, Summer-14, 18, Winter-17,18, Marks 7**
2. *Write a note on query evaluation plan.* **GTU : June-08, Marks 4**

5.2 Measuring Cost of Query

GTU : Dec.-10, Winter-12, Summer-13, Marks 7

- There are multiple possible evaluation plans for a query and it is important to be able to compare the alternatives in terms of their estimated cost and choose the best plan.
- There are many factors that contribute to query cost are -
  - Disk access
  - CPU time to execute the query
  - Cost of communication in distributed and parallel database system.
- The cost of access data from disk is an important cost. Normally disk access is relatively slow as compared to in-memory operations. Moreover, the CPU speed is much faster than the disk speed. Hence the time spent in disk is relatively dominant factor in query execution.
- Computing CPU access time is comparatively harder, hence we will not consider the CPU time to execute the query.
- Similarly the cost of communication does not matter for simple large databases present in the centralized database system.
- Typically disk access is the predominant cost and is also relatively easy to estimate, taking into account :
  - Number of seeks × Average-seek-cost
  - Number of blocks read × Average-block-read-cost
  - Number of blocks written × Average-block-write-cost
- Cost to write a block is greater than cost to read a block because data is read back after being written to ensure that the write was successful.
- We use **number of block transfers** from disk and **number of disk seeks** to estimate the query cost.

- Let,
  - b be the number to blocks
  - S be the number of seeks
  - t_T is average time required to transfer a block of data, in seconds.
  - t_S is average block access time in seconds.

  Then query cost can be computed using following formula $b*t_T+S*t_S$.

5.2.1 Algorithms for SELECT Operation

For selection operation, the file scan is an important activity. Basically file scan is a based on searching algorithms. These searching algorithms locate and retrieve the records that fulfills a selection condition.

Let us discuss various algorithms used for SELECT operation based in file scan.

Algorithm A1 : Linear search

- Scan each file block and test all records to see whether they satisfy the selection condition
  Cost = b_r block transfers + 1 seek

  Where,

  b_r denotes number of blocks containing records from relation r
- If selection is on a key attribute, can stop on finding record
  Cost = (b_r / 2) block transfers + 1 seek
- **Advantages of linear search**
  - Linear search works even-if there is no selection condition specified.
  - For linear search, there is no need to have records in the file in ordered form.
  - Linear search works regardless of indices.

Algorithm A2 : Binary search

- Applicable if selection is an equality comparison on the attribute on which file is ordered.
- Assume that the blocks of a relation are stored contiguously.
- Cost estimate is nothing but the number of disk blocks to be scanned.
  Cost of locating the first tuple by a binary search on the blocks = $\lceil \log_2(b_r) \rceil \times (t_T + t_S)$

  Where,

  b_r denotes number of blocks containing records from relation r.

  t_T is average time required to transfer a block of data, in seconds.

  t_S is average block access time, in seconds.

- If there are multiple records satisfying the selection add transfer cost of the number of blocks containing records that satisfy selection condition.

5.2.2 Join Strategies

- JOIN operation is the most time consuming operation to process.
- There are several different algorithms to implement joins
  1. Nested-loop join
  2. Block nested-loop join
  3. Indexed nested-loop join
  4. Merge-join
  5. Hash-join

  Choice of a particular algorithm is based on cost estimate.

(1) Algorithm for nested loop join

This algorithm is for computing $r \bowtie_\Theta s$

Let, r is called the outer relation and s the inner relation of the join.

```
for each tuple t_r in r do begin
    for each tuple t_s in s do begin
    test pair (t_r,t_s) to see if they satisfy the join condition θ
    if θ is satisfied, then, add (t_r,t_s) to the result.
    end
end
```

- This algorithm requires no indices and can be used with any kind of join condition.
- It is expensive since it examines every pair of tuples in the two relations.
- In the **worst case,** if there is enough memory only to hold one block of each relation, the estimated cost is $n_r \times b_s + b_r$ block transfers, plus $n_r + b_r$ seeks.
- If the smaller relation fits entirely in memory, use that as the inner relation.
  - Reduces cost to $b_r + b_s$ block transfers and 2 seeks.
- **For example -** Assume the query CUSTOMERS $\bowtie$ ORDERS (with join attribute only being CName)

  Number of records of customer : 10000 order : 5000

  Number of blocks of customer : 400 order : 100

  Formula Used :

  (1) $n_r \times b_s + b_r$ block transfers,

  (2) $n_r + b_r$ seeks

  r is outer relation and s is inner relation.

With order as outer relation :

$n_r = 5000$ $b_s = 400$, $b_r = 100$

$5000 \times 400 + 100 = 2000100$ block transfers and

$5000 + 100 = 5100$ seeks

With customer as the outer relation :

$n_r = 10000$, $b_s = 100$, $b_r = 400$

$10000 \times 100 + 400 = 1000400$ block transfers and

$10000 + 400 = 10400$ seeks

If smaller relation (order) fits entirely in memory, the cost estimate will be :

$b_r + b_s = 500$ block transfers

(2) Algorithm for block nested loop join

Variant of nested-loop join in which every block of inner relation is paired with every block of outer relation.

This algorithm is for computing $r \bowtie_\theta s$

Let, r is called the outer relation and s the inner relation of the join.

```
for each block B_r of r do
for each block B_s of s do
for each tuple t_r in B_r do begin
        for each tuple t_s in Bs do begin
            test pair (t_r , t_s) to see if they satisfy the join condition θ
            if θ is satisfied, then, add (t_r, t_s) to the result.
            end
        end
    end
end
```

Worst case estimate : $b_r \times b_s + b_r$ block transfers + $2 \times b_r$ seeks.

Each block in the inner relation s is read once for each block in the outer relation.

Best case : $b_r + b_s$ block transfers + 2 seeks

(3) Merge join

- In this operation, both the relations are sorted on their join attributes. Then merged these sorted relations to join them.
- Only equijoin and natural joins are used.
- The cost of merge join is,

  $b_r + b_s$ block transfers + $\lceil b_r / b_b \rceil + \lceil b_s / b_b \rceil$ seeks + The cost of sorting, if relations are unsorted.

(4) Hash join

- In this operation, the hash function h is used to partition tuples of both the relations.
- h maps A values to {0, 1, ..., n}, where A denotes the attributes of r and s used in the join.
- Cost of hash join is :

  $3(b_r + b_s) + 4 \times n$ block transfers + $2(\lceil b_r/b_b \rceil + \lceil b_s/b_b \rceil)$ seeks.

  If the entire build input can be kept in main memory no partitioning is required

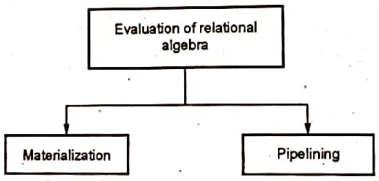
  cost estimate goes down to $b_r + b_s$.

Review Questions

1. *Explain linear search and binary search algorithm for selection operation.* **GTU : Dec.-10, Marks 7**
2. *Explain the measures of query cost, selection operation and join.* **GTU : Winter-12, Marks 7**
3. *Explain the measures of finding put cost of query processing.* **GTU : Summer-13, Marks 5**

5.3 Evaluation of Relational Algebra Expressions

- As we know, query consists of several relational algebra operators. Evaluation of query can be performed by evaluation of algebraic expressions.
- Evaluation of relational algebraic expression can be performed using two approaches -



(1) Materialization :

- The result of relational algebra operator operation is saved in some temporary or intermediate files which can be used for processing by next subsequent operation. Such an approach is called materialization.
- This approach is called materialization approach because - When the results are temporarily stored in intermediate files then it is said that the **tuples are materialized.**
- During evaluation of query a query tree is built. The process of materialization starts at the **lowest level** operation of the query tree. That means the lowest level expression is evaluated, the result is stored in intermediate files, this result is then used by subsequent higher level of the

tree, this process is repeated and finally the operation at the root of the query tree is evaluated giving the final result of the expression.

- The cost of materialization approach includes the cost of all operations as well as cost of writing results of each intermediate operation to disk.

(2) Pipelining :

- The process of sending the result of one operation to another operation without storing it to intermediate files is called **pipelining.**
- This approach **improves the performance** of the query.
- The efficiency of query evaluation can be improved by reducing the number of temporary files that are produced for storing intermediate results. The pipelining approach helps to improve the query evaluation process because it does not make use of temporary files and results are directly send to the next subsequent operator.

5.4 Query Equivalence

GTU : Dec.-09, Winter-13, Marks 3

Definition : Two relational algebra expressions are said to be equivalent if the two expressions generate the same set of tuples on every legal database instance.

The order of tuples is irrelevant.

An equivalence rule says that expressions of two forms are equivalent and can replace expression of first form by second or vice versa.

Equivalence rules

(1) **Conjunctive selection operations can be deconstructed into a sequence of individual selections.**

$\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$

(2) **Selection operations are commutative.**

$\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$

(3) **Only the last in a sequence of projection operations is needed, the others can be omitted.**

$\Pi_{L1}(\Pi_{L2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L1}(E)$

(4) **Selections can be combined with Cartesian products and theta joins.**

(i) $\sigma_{\theta}(E1 \times E2) = E1 \bowtie_{\theta} E2$

(ii) $\sigma_{\theta 1}(E1 \bowtie_{\theta 2} E2) = E1 \bowtie_{\theta 1 \wedge \theta 2} E2$

(5) **Theta-join operations (and natural joins) are commutative.**

$E1 \bowtie_{\theta} E2 = E2 \bowtie_{\theta} E1$

(6)

(a) Natural join operations are associative :

$(E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$

(b) Theta joins are associative :

$(E1 \bowtie_{\theta 1} E2) \bowtie_{\theta 2 \wedge \theta 3} E3 = E1 \bowtie_{\theta 1 \wedge \theta 3} (E2 \bowtie_{\theta 2} E3)$

where θ2 involves attributes from only E2 and E3

(7) The selection operation distributes over the theta join operation under the following two conditions :

(a) When all the attributes in θ0 involve only the attributes of one of the expressions (E1) being joined.

$\sigma_{\theta 0}(E1 \bowtie_{\theta} E2) = (\sigma_{\theta 0}(E1)) \bowtie_{\theta} E2$

(b) When θ_1 involves only the attributes of E1 and θ2 involves only the attributes of E2

$\sigma_{\theta 1 \wedge \theta 2}(E1 \bowtie_{\theta} E2) = (\sigma_{\theta 1}(E1)) \bowtie_{\theta} (\sigma_{\theta 2}(E2))$

(8) The projection operation distributes over the theta join operation as follows :

(a) if θ involves only attributes from L1 ∪ L2 :

$\Pi_{L1 \cup L2}(E1 \bowtie_{\theta} E2) = (\Pi_{L1}(E1)) \bowtie_{\theta} (\Pi_{L2}(E2))$

(b) Consider a join E1 $\bowtie_{\theta}$ E2 :

- Let L1 and L2 be sets of attributes from E1 and E2 , respectively.
- Let L3 be attributes of E1 that are involved in join condition θ, but are not in L1 ∪ L2 and
- Let L4 be attributes of E2 that are involved in join condition θ, but are not in L1 ∪ L2.

$\Pi_{L1 \cup L2}(E1 \bowtie_{\theta} E2) = (\Pi_{L1 \cup L2}((\Pi_{L1 \cup L3}(E1)) \bowtie_{\theta} (\Pi_{L2 \cup L4}(E2)))$

(9) The set operation union and intersection are commutative

$E1 \cup E2 = E2 \cup E1$

$E1 \cap E2 = E2 \cap E1$

(10) The set union and intersection are associative

$(E1 \cup E2) \cup E3 = E1 \cup (E2 \cup E3)$

$(E1 \cap E2) \cap E3 = E1 \cap (E2 \cap E3)$

(11) The selection operation distributes over union ∪, ∩ and -

$\sigma_{\theta}(E1 - E2) = \sigma_{\theta}(E1) - \sigma_{\theta}(E2)$

and similarly for ∪ and ∩ in place of -

$\sigma_{\theta}(E1 - E2) = \sigma_{\theta}(E1) - E2$

and similarly for ∩ in place of –, but not for ∪

(12) The projection operation distributes over union

$\Pi_L(E1 \cup E2) = (\Pi_L(E1)) \cup (\Pi_L(E2))$

For example - Consider the following query

Find the names of all employees in accounts department, along with the department location in which the employee work for.

The schema is

Employee(empID, empname, deptname, salary)
Works_for(empID, deptID,working_hr)
Department(deptID, deptname,deptloc)

The relational algebra is -

$\Pi_{ename,\ deptloc}(\sigma_{deptname="Account"}(employee \bowtie (Works\_for \bowtie \Pi_{deptID,\ deptloc}(dept))))$

Now applying transformation using 7(a) rule we get

$\Pi_{ename,\ deptloc}((\sigma_{deptname="Account"}(employee)) \bowtie (Works\_for \bowtie \Pi_{deptID,\ deptloc}(dept)))$

Performing selection as early as possible reduces the size of the relation to be joined.

Review Question

1. *Explain evaluation of expression process in query optimization.*

GTU : Dec.-09, Marks 4, Winter-13, Marks 3

5.5 Query Optimization Algorithms

GTU : Summer-17, Marks 3

5.5.1 Heuristic Estimation

- Heuristic is a rule that leads to least cost in most of cases.
- Systems may use heuristics to reduce the number of choices that must be made in a cost-based fashion.
- Heuristic optimization transforms the query-tree by using a set of rules that typically t improve execution performance. These rules are
  1. Perform selection early (reduces the number of tuples).
  2. Perform projection early (reduces the number of attributes).
  3. Perform most restrictive selection and join operations before other similar operations (such as cartesian product).
- Some systems use only heuristics, others combine heuristics with partial cost-based optimization.

Steps in heuristic estimation

Step 1 : Scanner and parser generate initial query representation.

Step 2 : Representation is optimized according to heuristic rules.

Step 3 : Query execution plan is developed.

For example : Suppose there are two relational algebra -

(1) $\sigma_{city="Pune"}(\pi_{cname}$ Branch)⋈ Account ⋈ Customer)

(2) $\pi_{cname}(\sigma_{city="Pune"}$(Branch⋈ Account ⋈ Customer))

The query evaluation plan can be drawn using the query trees as follows -

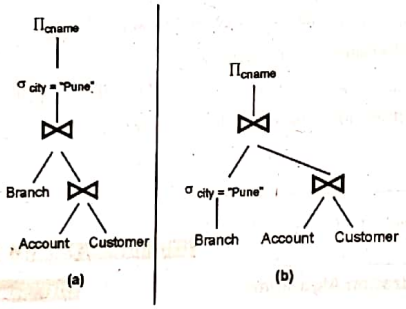


Fig. 5.5.1 Query evaluation plan

Out of the above given query evaluation plans, the Fig. 5.5.1 (b) is much faster than Fig. 5.5.1 (a) because - in Fig. 5.5.1 (a) the join operation is among branch, account and customer, whereas in Fig. 5.5.1 (b) the join of (account and customer) is made with the selected tuple for City="Pune". Thus the output of entire table for join operation is much more than the join for some selected tuples. Thus we get choose the optimized query.

5.5.2 Cost Based Estimation

- A cost based optimizer will look at all of the possible ways or scenarios in which a query can be executed.
- Each scenario will be assigned a 'cost', which indicates how efficiently that query can be run.
- Then, the cost based optimizer will pick the scenario that has the least cost and execute the query using that scenario, because that is the most efficient way to run the query.

- Scope of query optimization is a query block. Global query optimization involves multiple query blocks.
- Cost components for query execution
  - Access cost to secondary storage
  - Disk storage cost
  - Computation cost
  - Memory usage cost
  - Communication cost
- Following information stored in DBMS catalog and used by optimizer
  - File size
  - Organization
  - Number of levels of each multilevel index
  - Number of distinct values of an attribute
  - Attribute selectivity
- RDBMS stores histograms for most important attributes.

Review Question

1. *Explain heuristics in optimization.* **GTU. : Summer-17, Marks 3**

5.6 Oral Questions and Answers

Q.1 What is the need for query optimization?

Ans. : Query optimization is required for fast execution of long running complex queries.

Q.2 Which cost component are used most commonly as the basis for cost function ?

Ans. : Disk access or secondary storage access is considered most commonly as a basis for cost function.

Q.3 What is query execution plan ?

Ans. : To specify fully how to evaluate a query, we need not only to provide the relational-algebra expression, but also to annotate it with instructions specifying how to evaluate each operation. This annotated structure is called query execution plan.

Q.4 What are two approaches of evaluation of relational algebra expression?

Ans. : The two approaches of evaluation of relational algebra expression are –

(1) Materialization (2) Pipelining

Q.5 What is query equivalence ?

Ans. : Two relational algebra expressions are said to be equivalent if the two expressions generate the same set of tuples on every legal database instance.

Q.6 Give any two query equivalence rules.

Ans. : (1) Selection operations are commutative.

(2) Selections can be combined with Cartesian products and theta joins.

(3) Natural join operations are associative.

❑❑❑

6 Storage Strategies

Syllabus

DIndices, B-trees, hashing.

Contents

6.1 Indices

- An index is a **data structure that organizes data records** on the disk to make the **retrieval** of data efficient.
- The search key for an index is collection of one or more fields of records using which we can efficiently retrieve the data that satisfy the search conditions.
- The indexes are required to speed up the search operations on file of records.
- There are two types of indices -
  - **Ordered Indices :** This type of indexing is based on sorted ordering values.
  - **Hash Indices :** This type of indexing is based on uniform distribution of values across range of buckets. The address of bucket is obtained using the hash function.
- There are several techniques of for using indexing and hashing. These techniques are evaluated based on following factors -
  - **Access Types :** It supports various types of access that are supported efficiently.
  - **Access Time :** It denotes the time it takes to find a particular data item or set items.
  - **Insertion Time :** It represents the time required to insert new data item.
  - **Deletion Time :** It represents the time required to delete the desired data item.
  - **Space overhead :** The space is required to occupy the index structure. But allocating such extra space is worth to achieve improved performance.

Example 6.1.1 *Since indices speed query processing, why might they not be kept on several search keys ? List as many reasons as possible.*

Solution : Reasons for not keeping several search indices are :

a. Every index requires additional CPU time and disk I/O overhead during inserts and deletions.

b. Indices on non-primary keys might have to be changed on updates, although an index on the primary key might not (as updates typically do not modify the primary key attributes).

c. Each extra index requires additional storage space.

d. For queries which involve conditions on several search keys, efficiency might not be bad even if only some of the keys have indices on them.

Therefore database performance is improved less by adding indices when many indices already exist.

6.1.1 Ordered Indices

6.1.1.1 Primary and Clustered Indices

Primary Index :

- An index on a set of fields that includes the primary key is called a primary index. The primary index file should be always in sorted order.
- The primary indexing is always done when the data file is arranged in sorted order and primary indexing contains the primary key as its search key.
- Consider following scenario in which the primary index consists of few entries as compared to actual data file.

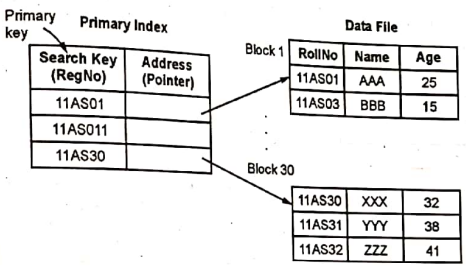


Fig. 6.1.1 : Example of primary index

- Once if you are able to locate the first entry of the record containing block, other entries are stored continuously. For example if we want to search a record for RegNo 11AS32 we need not have to search for the entire data file. With the help of primary index structure we come to know the location of the record containing the RegNo 11AS30, now when the first entry of block 30 is located, then we can easily locate the entry for 11AS32.
- We can apply binary search technique. Suppose there are n = 300 blocks in a main data file then the number of accesses required to search the data file will be $\log_2 n + 1 = (\log_2 300) + 1 \approx 9$
- If we use primary index file which contains at the most n = 3 blocks then using binary search technique, the number of accesses required to search using the primary index file will be $\log_2 n + 1 = (\log_2 3) + 1 \approx 3$
- This shows that using primary index the access time can be deduced to great extent.

Clustered Index :

- In some cases, the index is created on non-primary key columns which may not be unique for each record. In such cases, in order to identify the records faster, we will group two or more columns together to get the unique values and create index out of them. This method is known as **clustering index**.

- When a file is organized so that the ordering of data records is the same as the ordering of data entries in some index then say that index is **clustered**, otherwise it is an **unclustered index**.
- Note that, the data file need to be in sorted order.
- Basically, records with similar characteristics are grouped together and indexes are created for these groups.
- **For example**, students studying in each semester are grouped together. i.e.; 1st semester students, 2nd semester students, 3rd semester students etc. are grouped.

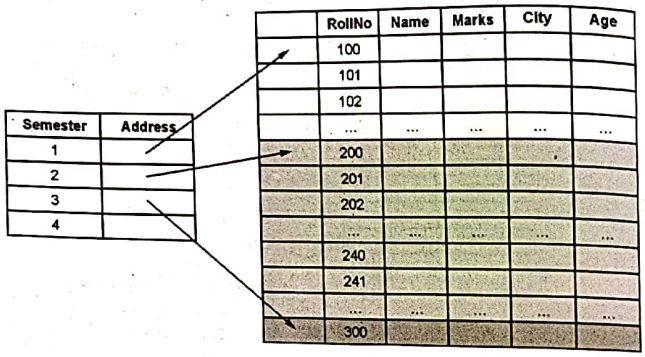


Fig. 6.1.2 Clustered Index

6.1.1.2 Dense and Sparse Indices

There are two types of ordered indices :

1) Dense index :

- An index record appears for every search key value in file.
- This record contains search key value and a pointer to the actual record.
- For example :

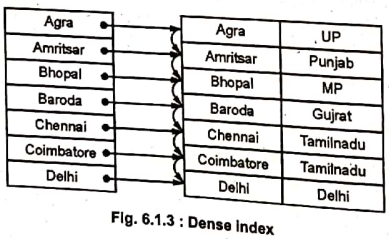


Fig. 6.1.3 : Dense Index

2) Sparse index :

- Index records are created only for some of the records.
- To locate a record, we find the index record with the largest search key value less than or equal to the search key value we are looking for.
- We start at that record pointed to by the index record, and proceed along the pointers in the file (that is, sequentially) until we find the desired record.
- For example –

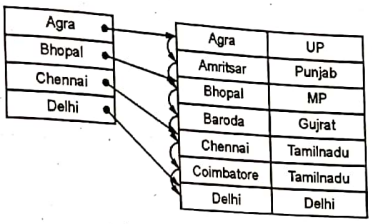


Fig.6.1.4 : Sparse index

6.1.1.3 Single and Multilevel Indices

Single level indexing :

- A single-level index is an auxiliary file that makes it more efficient to search for a record in the data file.
- The index is usually specified on one field of the file (although it could be specified on several fields).
- Each index can be in the following form.

Search Key	Pointer to Record

- The index file usually occupies considerably less disk blocks than the data file because its entries are much smaller.
- A binary search on the index yields a pointer to the file record.
- The types of single level indexing can be primary indexing, clustering index or secondary indexing.

- Example : Following Fig. 6.1.5 represents the single level indexing –

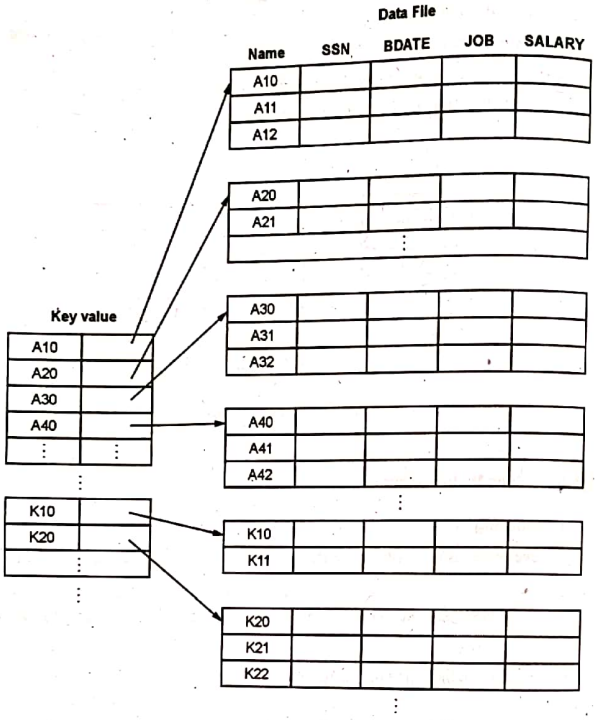


Fig. 6.1.5 : Single level indexing

Multilevel Indexing :

- There is an immense need to keep the index records in the main memory so as to speed up the search operations. If single-level index is used, then a large size index cannot be kept in memory which leads to multiple disk accesses.
- Multi-level index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.

- The multilevel indexing can be represented by following Fig. 6.1.6

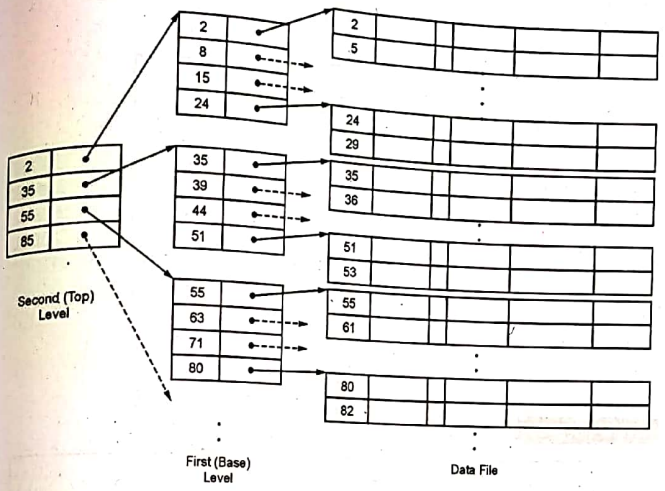


Fig. 6.1.6 Multilevel indexing

6.1.1.4 Secondary Indices

- In this technique two levels of indexing are used in order to reduce the mapping size of the first level.
- Initially, for the first level, a large range of numbers is selected so that the mapping size is small. Further, each range is divided into subsequent sub ranges.
- It is used to optimize the query processing and access records in a database with some information other than the usual search key.

- For example -

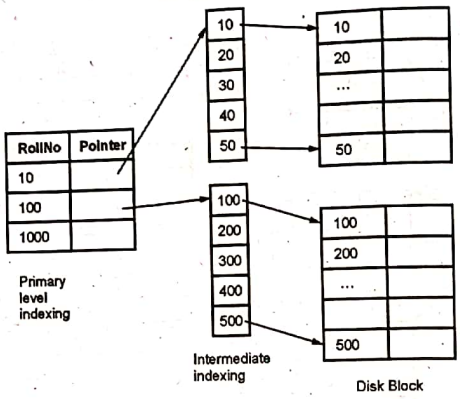


Fig. 6.1.7 Secondary Indexing

Review Questions

1. *Illustrate indexing with suitable example*
2. *What is the use of an index structure and explain the concept of ordered indices.*

6.2 B-Trees

- B-tree is a specialized multiway tree used to store the records in a disk.
- There are number of subtrees to each node. So that the height of the tree is relatively small. So that only small number of nodes must be read from disk to retrieve an item.
- The **goal** of B-trees is to get fast access of the data.

Structure of B+ Tree

- The typical node structure of B+ node is as follows -

P_1	K_1	P_2	K_2	...	P_{n-1}	K_{n-1}	P_n

It contains up to n − 1 search-key values $K_1, K_2, \ldots, K_{n-1}$, and n pointers $P_1, P_2, \ldots, P_n$.
The search-key values within a node are kept in sorted order; thus, if i < j, then $K_i < K_j$.

- A B-tree allows search-key values to appear only once (if they are unique).

- **For example -**

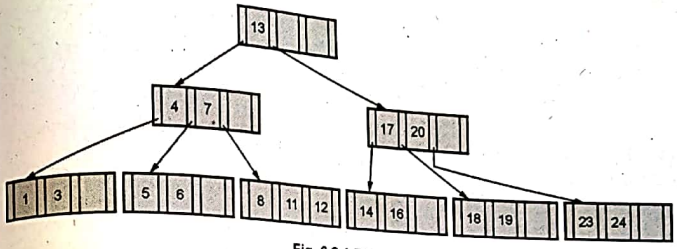


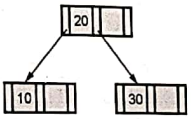
Fig. 6.2.1 B-Tree

Example 6.2.1 *Create B tree of order 3 for following data: 20,10,30,15,12,40,50.*

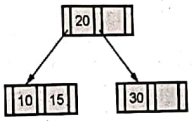
Solution : The B tree of order 3 means at the most two key values are allowed in each node of B-Tree.

Step 1 : Insert 20, 10 in ascending order

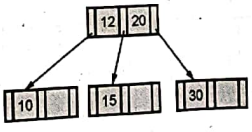
Step 2 : If we insert the value 30. The sequence becomes 10,20,30. As only two key values are allowed in each node (being order 3), the 20 will go up.



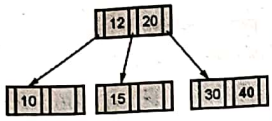
Step 3 : Now insert 15.



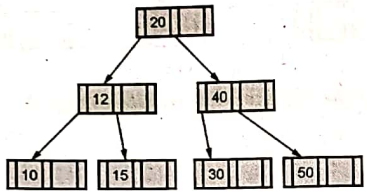
Step 4 : Insert 12. The sequence becomes 10, 12, 15. The middle element 12 will go up.



Step 5 : Insert 40



Step 6 : Insert 50. The sequence becomes 30,40,50. The 40 will go up. But again it forms 12,20,40 sequence and then 20 will go up. Thus the final B Tree will be,



This is the final B-Tree

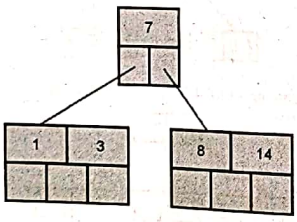
Example 6.2.2 *Construct a B-tree of order 5 for the following data*
3,14,7,1,8,5,11,17,13,6,23,12,20,26,4,16,18,24,25,19

Solution : The order 5 means at the most 4 keys are allowed. The internal node should have at least 3 non empty children and each leaf node must contain at least 2 keys.

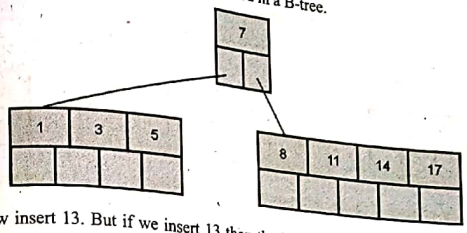
Step 1 : Insert 3, 14, 7, 1 as follows

Step 2 : If we insert 8 then we need to split the node 1, 3, 7, 8, 14 at medium.

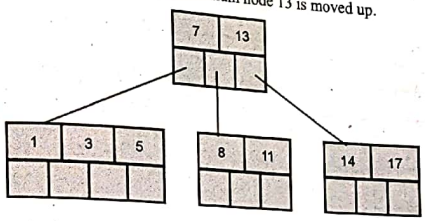
Hence



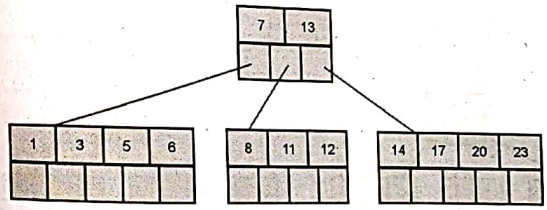
Step 3 : Insert 5, 11, 7 which can be easily inserted in a B-tree.



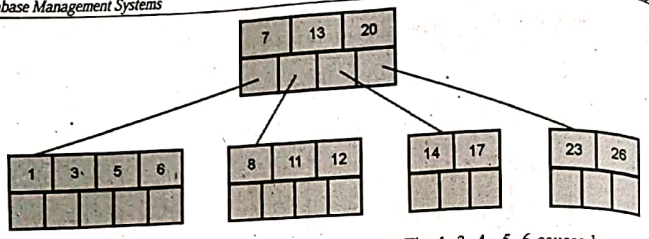
Step 4 : Now insert 13. But if we insert 13 then the leaf node will have 5 keys which is not allowed. Hence 8, 11, 13 , 14, 17 is split and medium node 13 is moved up.



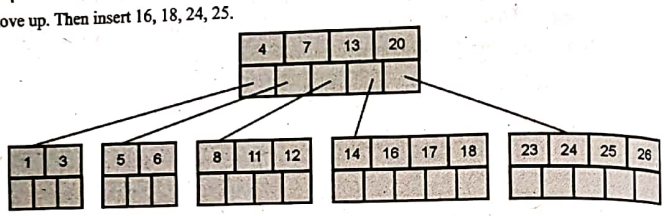
Step 5 : Now insert 6, 23, 12, 20 without any split.



Step 6 : The 26 is inserted to the rightmost leaf node. Hence 14, 17, 20 , 23, 26 the node is split and 20 will be moved up.

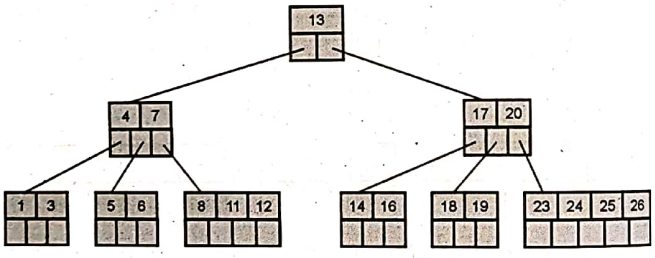


Step 7 : Insertion of node 4 causes left most node to split. The 1, 3, 4 , 5, 6 causes key 4 to move up. Then insert 16, 18, 24, 25.



Step 8 : Finally insert 19. Then 4, 7, 13, 19, 20 needs to be split. The median 13 will be moved up to form a root node.

The tree then will be -



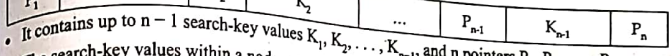
Thus the B-tree is constructed.

6.2.1 B+ Trees

- The B+ tree is similar to binary search tree. It is a balanced tree in which the internal nodes direct the search.
- The leaf nodes of B+ trees contain the **data entries.**

Structure of B+ Tree

- The typical node structure of B+ node is as follows -



- It contains up to n − 1 search-key values $K_1, K_2, \ldots, K_{n-1}$, and n pointers $P_1, P_2, \ldots, P_n$.
- The search-key values within a node are kept in sorted order; thus, if i < j, then $K_i < K_j$.
- To retrieve all the leaf pages efficiently we have to link them using **page pointers.** The sequence of leaf pages is also called as **sequence set.**
- Following Fig. 6.2.2 represents the example of B+ tree.

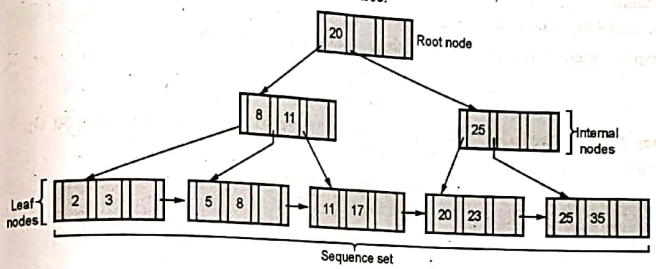


Fig. 6.2.2 B+ Tree

- The B+ tree is called dynamic tree because the tree structure can grow on insertion of records and shrink on deletion of records.

Characteristics of B+ Tree

Following are the characteristics of B+ tree :

1) The B+ tree is a balanced tree and the operations insertions and deletion keeps the tree balanced.
2) A minimum occupancy of 50 percent is guaranteed for each node except the root.
3) Searching for a record requires just traversal from the root to appropriate leaf.

6.2.1.1 Insertion Operation

Algorithm for insertion :

Step 1 : Find correct leaf L.

Step 2 : Put data entry onto L.

i) If L has enough space, done!

ii) Else, must split L (into L and a new node L2)

- Allocate new node
- Redistribute entries evenly
- Copy up middle key.

- Insert index entry pointing to L2 into parent of L.

Step 3 : This can happen recursively

i) To split index node, redistribute entries evenly, but push up middle key. (Contrast with leaf splits.)

Step 4 : Splits "grow" tree; root split increases height.

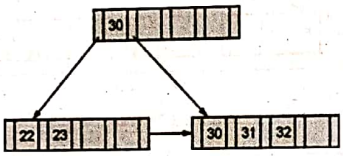
i) Tree growth: gets wider or one level taller at top.

Example 6.2.3 *Construct B+ tree for following data. 30,31,23,32,22,28,24,29, where number of pointers that fit in one node are 5.*

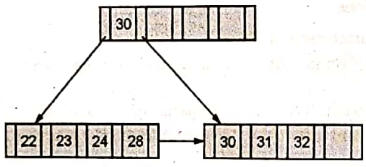
Solution : In B+ tree each node is allowed to have the number of pointers to be 5. That means at the most 4 key values are allowed in each node.

Step 1 : Insert 30,31,23,32. We insert the key values in ascending order.

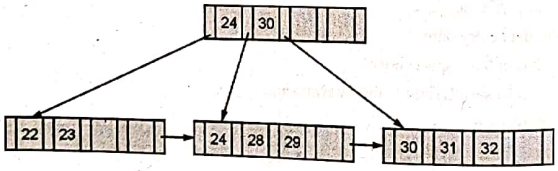
Step 2 : Now if we insert 22, the sequence will be 22 , 23, 30, 31, 32. The middle key 30, will go up.



Step 3 : Insert 28,24. The insertion is in ascending order.



Step 4 : Insert 29. The sequence becomes 22, 23, 24, 28, 29. The middle key 24 will go up. Thus we get the B+ tree.

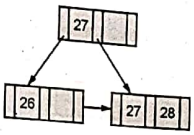


Example 6.2.4 *Construct B+ tree to insert the following (order of the tree is 3) 26,27,28,3,4,7,9,46,48,51,2,6*

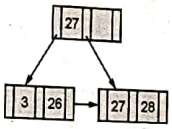
Solution : Order means maximum number of children allowed by each node. Hence order 3 means at the most 2 key values are allowed in each node.

Step 1 : Insert 26, 27 in ascending order

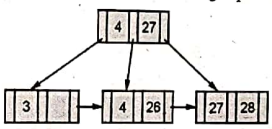
Step 2 : Now insert 28. The sequence becomes 26,27,28. As the capacity of the node is full, 27 will go up. The B+ tree will be,



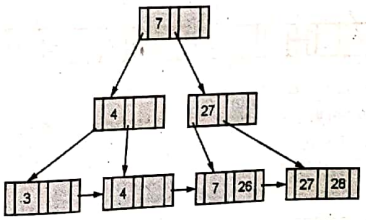
Step 3 : Insert 3. The partial B+ Tree will be,



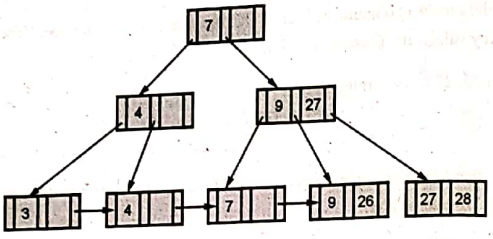
Step 4 : Insert 4. The sequence becomes 3,4, 26. The 4 will go up. The partial B+ tree will be -



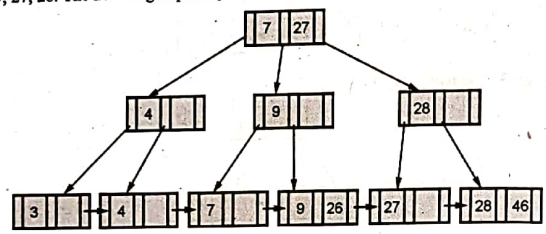
Step 5 : Insert 7. The sequence becomes 4,7,26. The 7 will go up. Again from 4,7,27 the 7 will go up. The partial B+ Tree will be,



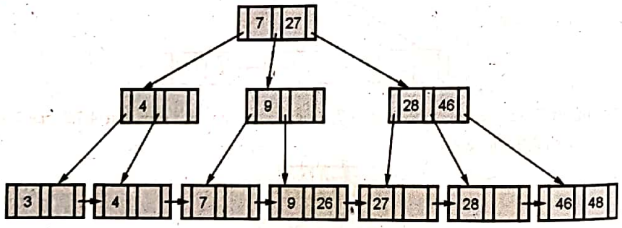
Step 6 : Insert 9. By inserting 7,9, 26 will be the sequence. The 9 will go up. The partial B+ tree will be,



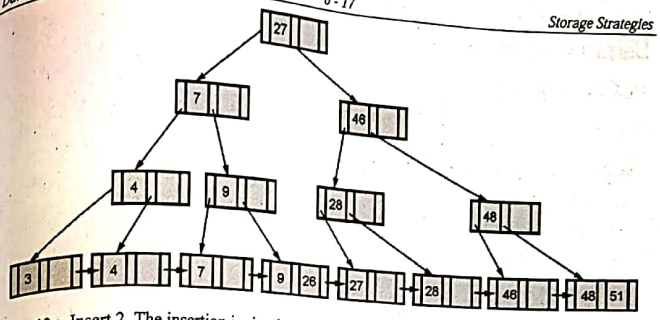
Step 7 : Insert 46. The sequence becomes 27,28,46. The 28 will go up. Now the sequence becomes 9, 27, 28. The 27 will go up and join 7. The B+ Tree will be,



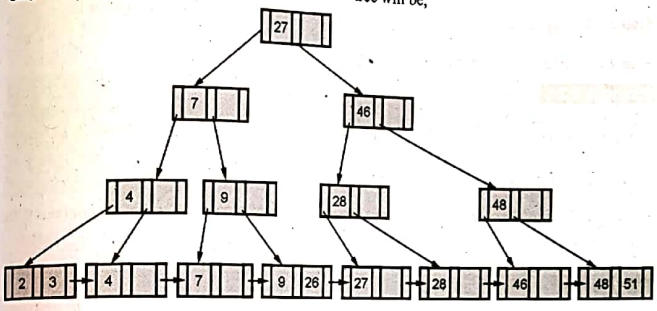
Step 8 : Insert 48. The sequence becomes 28,46,48. The 46 will go up. The B+ Tree will become,



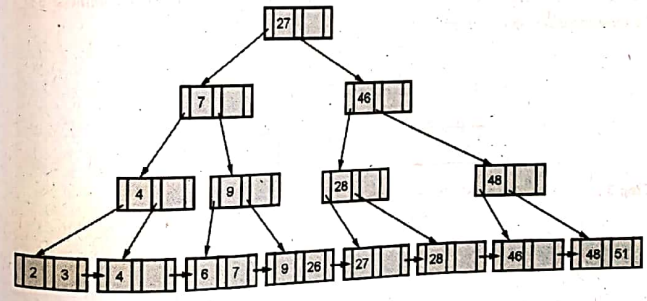
Step 9 : Insert 51. The sequence becomes 46,48,51. The 48 will go up. Then the sequence becomes 28, 46, 48. Again the 46 will go up. Now the sequence becomes 7,27, 46. Now the 27 will go up. Thus the B+ tree will be



Step 10 : Insert 2. The insertion is simple. The B+ tree will be,



Step 11 : Insert 6. The insertion can be made in a vacant node of 7(the leaf node). The final B+ tree will be,



6.2.1.2 Deletion Operation

Algorithm for deletion :

Step 1 : Start at root, find leaf L with entry, if it exists.

Step 2 : Remove the entry.

i) If L is at least half-full, done!

ii) If L has only d-1 entries,

- Try to **re-distribute**, borrowing keys from sibling.
  (adjacent node with same parent as L).
- If redistribution fails, **merge L and sibling.**

Step 3 : If merge occurred, must delete entry (pointing to L or sibling) from parent of L.

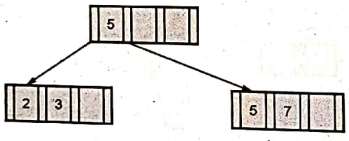
Step 4 : Merge could propagate to root, decreasing height.

Example 6.2.5 *Construct B+ Tree for the following set of key values (2,3,5,7,11,17,19,23,29,31) Assume that the tree is initially empty and values are added in ascending order. Construct B+ tree for the cases where the number of pointers that fit one node is four. After creation of B+ tree perform following series of operations :*
(a) Insert 9. (b) Insert 10. (c) Insert 8. (d) Delete 23. (e) Delete 19.

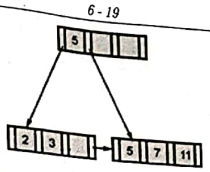
Solution : The number of pointers fitting in one node is four. That means each node contains at the most three key values.

Step 1 : Insert 2, 3, 5.

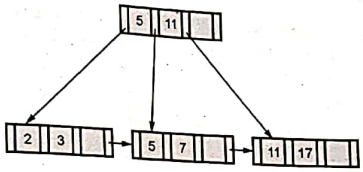
Step 2 : If we insert 7, the sequence becomes 2, 3, 5, 7. Since each node can accommodate at the most three key, the 5 will go up, from the sequence 2, 3, 5 , 7.



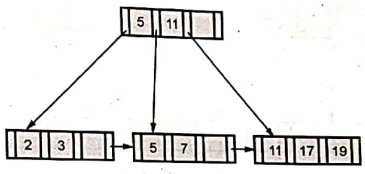
Step 3 : Insert 11. The partial B+ tree will be,



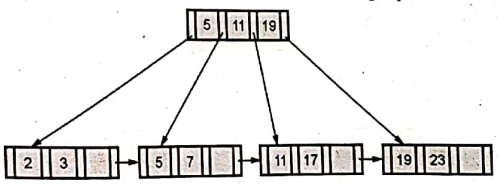
Step 4 : Insert 17. The sequence becomes 5,7, 11,17. The element 11 will go up. Then the partial B+ tree becomes,



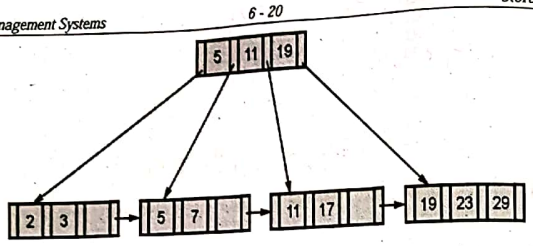
Step 5 : Insert 19.



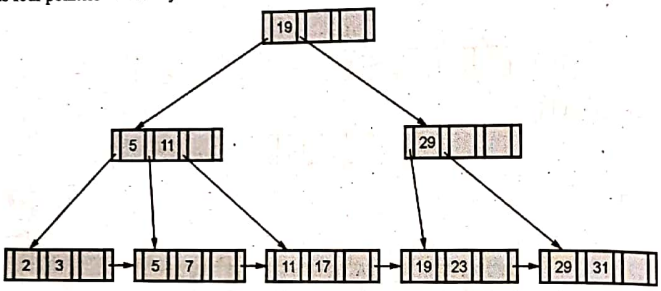
Step 6 : Insert 23. The sequence becomes 11,17,19,23. The 19 will go up.



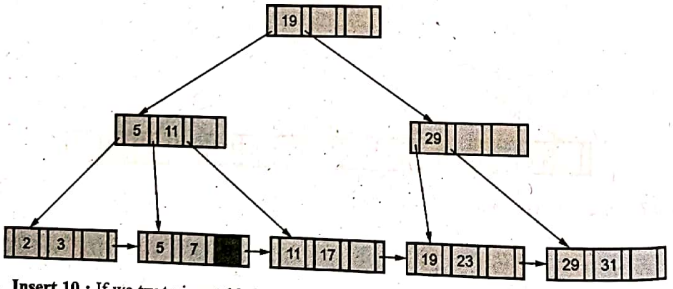
Step 7 : Insert 29. The partial B+ tree will be,



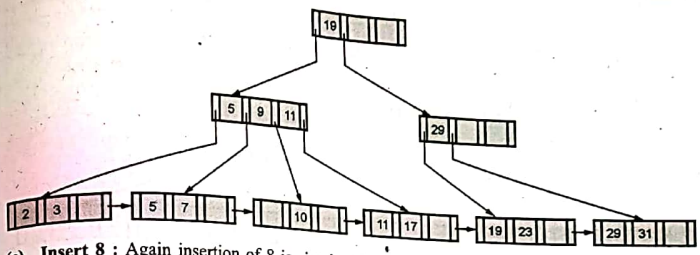
Step 8 : Insert 31. The sequence becomes 19,23,29, 31. The **29 will go up**. Then at the upper level the sequence becomes 5,11,19,29. Hence again **19 will go up** to maintain the capacity of node (it is four pointers= three key values at the most). Hence the complete B+ tree will be,



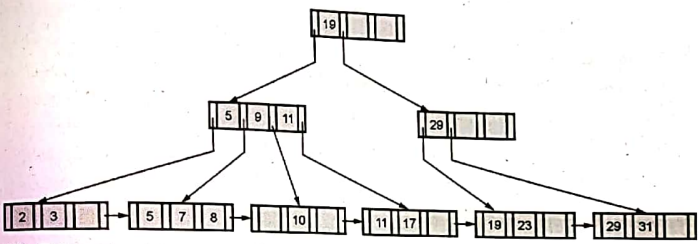
(a) Insertion of 9 : It is very simple operation as the node containing 5,7 has one space vacant to accommodate. The B+ tree will be,



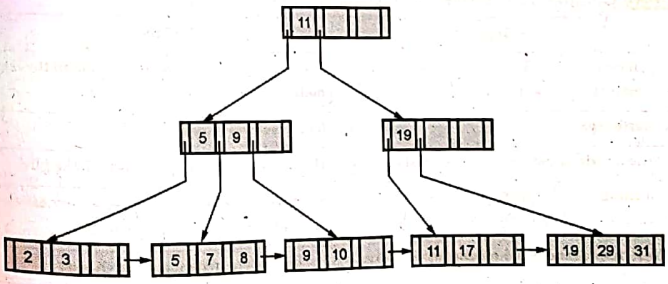
(b) Insert 10 : If we try to insert 10 then the sequence becomes 5,7,9,10. The **9 will go up.** The B+ tree will then become -



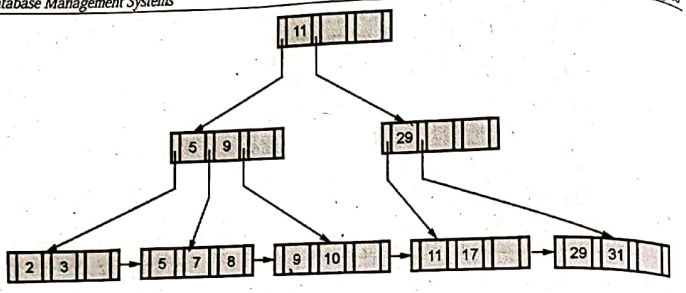
(c) Insert 8 : Again insertion of 8 is simple. We have a vacant space at node 5,7. So we just insert the value over there. The B+ tree will be-



(d) Delete 23 : Just remove the key entry of 23 from the node 19,23. Then **merge the sibling node** to form a node 19,29,31. Get down the entry of 11 to the leaf node. Attach the node of 11,17 as a left child of 19.



(e) Delete 19 : Just delete the entry of 19 from the node 19,29,31. Delete the internal node key 19. Copy the 29 up as an internal node as it is an in order successor node.



Merits of B+ Index Tree Structure

1. In B+ tree the data is stored in leaf node so searching of any data requires scanning only of leaf node alone.
2. Data is ordered in linked list.
3. Any record can be fetched in equal number of disk accesses.
4. Range queries can be performed easily as leaves are linked up.
5. Height of the tree is less as only keys are used for indexing.
6. Supports both random and sequential access.

Demerits of B+ Index Tree Structure

1. Extra insertion of non leaf nodes.
2. There is space overhead.

6.2.2 Comparison between B Tree and B+ Trees

B Trees	B+ Trees
In a B-tree you can store both **keys** and **data** in the **internal and leaf nodes.**	In a B+ tree you have to store the **data in the leaf nodes** only.
It wastes space.	**It does not waste space.**
The leaf node **cannot** store using **linked list.**	The leaf nodes are connected using linked list.
Searching becomes **difficult** in B-tree as data cannot be found in the leaf node.	**Searching** of any data in a B+ tree is very **easy** because all data is found in leaf nodes.
The B-tree does **not store redundant search key.**	The B-tree stores **redundant search key.**

Review Questions

1. *Write down detailed notes on ordered indices and B-tree index files*
2. *Explain the B+ tree indexes on multiple keys with suitable example.*
3. *What are the merits and demerits of B+ tree index structures ?*

6.3 Hashing

- Hash file organization method is the one where data is stored at the data blocks whose address is generated by using **hash function.**
- The memory location where these records are stored is called as **data block** or **bucket**. This bucket is capable of storing one or more records.
- The hash function can use any of the column value to generate the address. Most of the time, hash function uses **primary key** to generate the hash index - address of the data block.
- Hash function can be **simple mathematical function** to any complex mathematical function.
- For example - Following Fig. 6.3.1 represents the records of student can be searched using hash based indexing. In this example the hash function is based on the **age** field of the record. Here the index is made up of data entry k* which is **actual data record**. For a hash function the **age** is converted to binary number format and the last two digits are considered to locate the student record.

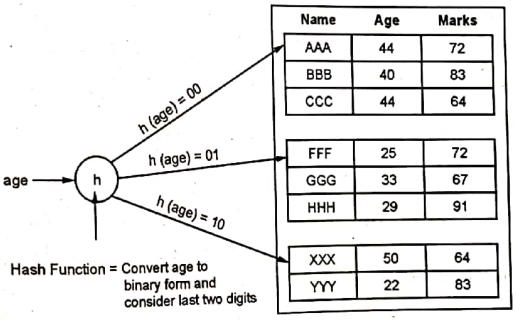


Fig. 6.3.1 : Hash based indexing

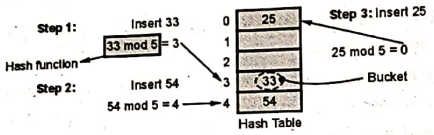
6.3.1 Basic Terms used in Hashing

1) **Hash Table :** Hash table is a data structure used for storing and retrieving data quickly. Every entry in the hash table is made using Hash function.

2) Hash function :

- Hash function is a function used to place data in hash table.
- Similarly hash function is used to retrieve data from hash table.
- Thus the use of hash function is to implement hash table.

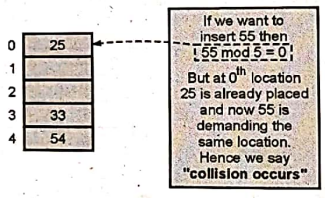
For example : Consider hash function **as key** **mod 5**. The hash table of size 5.



3) Bucket : The hash function H(key) is used to map several dictionary entries in the hash table. Each position of the hash table is called bucket.

4) Collision : Collision is situation in which hash function returns the same address for more than one record.

For example :



5) Probe : Each calculation of an address and test for success is known as a probe.

6) Synonym : The set of keys that has to the same location are called synonyms. For example - In above given hash table computation 25 and 55 are synonyms.

7) Overflow : When hash table becomes full and new record needs to be inserted then it is called overflow.

For example -

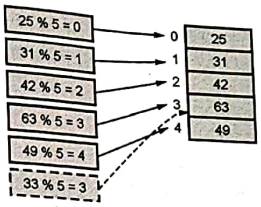


Fig. 6.3.2 : Overflow situation

Review Question

1. *Write short note on – hashing*

6.4 Static Hashing

- In this method of hashing, the resultant data **bucket address will be always same.**

Index	Stud_RollNo
0	34780
1	34781
2	34782
3	34783
4	34784
5	34785
6	34786
7	34787
8	34788
9	34789

- That means, if we want to generate address for **Stud_RollNo = 34789**. Here if we use mod 10 hash function, it always result in the same bucket address 9. There will not be any changes to the bucket address here.
- Hence number of data buckets in the memory for this static hashing remains constant throughout. In our example, we will have ten data buckets in the memory used to store the data.
- If there is **no space** for some data entry then we can allocate **new overflow page**, put the data record onto that page and add the page to **overflow chain** of the bucket. For example if we want to add the **Stud_RollNo = 35111** in above hash table then as there is no space for this entry and the hash address indicate to place this record at index 1, we create overflow chain as shown in Table 6.4.1.

Index	Stud_RollNo
0	34780
1	34781 → 35111
2	34782
3	34783
4	34784
5	34785
6	34786
7	34787
8	34788
9	34789

Table 6.4.1

Advantages of Static Hashing

(1) It is simple to implement.

(2) It allows speedy data storage.

Disadvantages of Static Hashing

There are two major disadvantages of static hashing :

1) In static hashing, there are fixed number of buckets. This will create a problematic situation if the number of records grow or shrink.

2) The ordered access on hash key makes it inefficient.

Example 6.4.1 *Why is hash structure not the best choice for a search key on which range of queries are likely ?*

Solution :

- A range query cannot be answered efficiently using a hash index, we will have to read all the buckets.
- This is because key values in the range do not occupy consecutive locations in the buckets, they are distributed uniformly and randomly throughout all the buckets.

6.4.1 Open Hashing

The **open hashing** is a form of **static hashing** technique. When the collision occurs, that means if the hash key returns the same address which is already allocated by some data record, then the next **available data block** is used to enter new record instead of overwriting the old record. This technique is also called as **linear probing**. For example

Consider insertion of record 105 in the hash table below with the hash function h (key) mod 10.

Index	Stud_RollNo
0	10
1	1
2	22
3	
4	
5	55
6	106
7	
8	88
9	19

The 105 is probed at next empty data block as follows -

Index	Stud_RollNo
0	10
1	1
2	22
3	
4	
5	55
6	106
7	105
8	88
9	19

Advantages :

1) It is **faster** technique. 2) It is **simple** to implement.

Disadvantages :

1) It forms clustering, as the record is just inserted to next free available slot.
2) If the hash table gets full then the next subsequent records can not be accommodated.

6.5 Dynamic Hashing

- The problem with static hashing is that it does not expand or shrink dynamically as the size of the database grows or shrinks.
- Dynamic hashing provides a mechanism in which data buckets are added and removed dynamically and on-demand.
- The most commonly used technique of dynamic hashing is extendible hashing.

6.5.1 Extendible Hashing

The extendible hashing is a dynamic hashing technique in which, if the bucket is overflow, then the number of buckets are doubled and data entries in buckets are re-distributed.

Example of extendible hashing :

In extendible hashing technique the **directory** of pointers to bucket is used. Refer following Fig. 6.5.1.

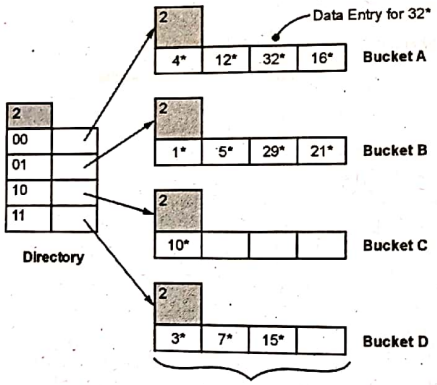
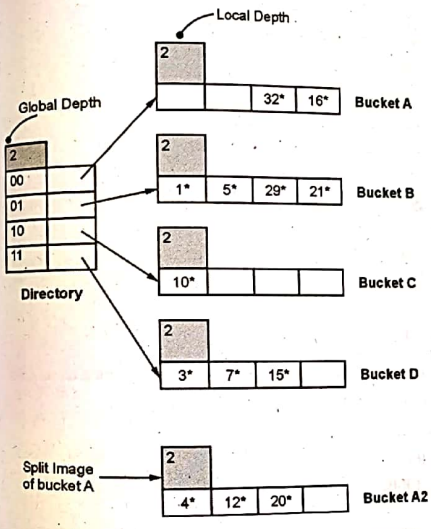


Fig. 6.5.1 Extendible hashing

To locate a data entry, we apply a hash function to search the data we us **last two digits of binary representation of number**. For instance binary representation of 32* = 10000000. The last two bits are 00. Hence we store 32* accordingly.

Insertion operation :

- Suppose we want to insert 20* (binary 10100). But with 00, the bucket A is full. So we must **split the bucket** by allocating new bucket and redistributing the contents, across the old bucket and its split image.
- For splitting, we consider last three bits of h(r).
- The redistribution while insertion of 20* is as shown in following Fig. 6.5.2



Decimal Value	Binary Value
32	100000
16	10000
1	0001
5	0101
29	011101
21	010101
10	01010
3	011
7	0111
15	01111
4	100

Fig. 6.5.2 : During insertion process

The split image of bucket A i.e. A2 and old bucket A are based on last two bits i.e. 00. Here we need two data pages, to adjacent additional data record. Therefore here it is necessary to double the directory using three bits instead of two bits. Hence,

- There will be binary versions for buckets A and A2 as 000 and 100.

- In extendible hashing, last bits d is called **global depth** for directory and d is called **local depth** for data pages or buckets. After insetion of 20*, the global depth becomes 3 as we consider last three bits and local depth of A and A2 buckets become 3 as we are considering last three bits for placing the data records. Refer Fig. 6.5.3.

(**Note :** Student should refer binary values given in Fig. 6.5.2, for understanding insertion operation)

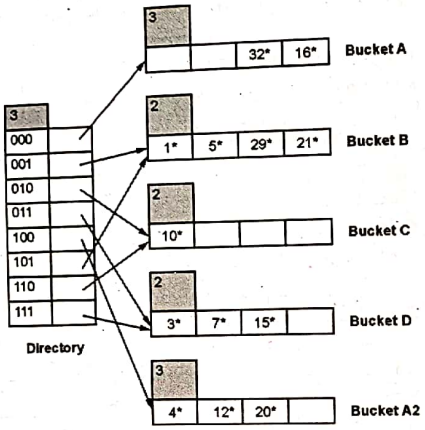


Fig. 6.5.3 : After insertion of 20*

- Suppose if we want to insert 11*, it belongs to bucket B, which is already full. Hence let us split bucket B into old bucket B and split image of B as B2.
- The local depth of B and B2 now becomes 3.
- Now for bucket B, we get and 1 = **001**
  11 = **10001**
- For bucket B2, we get
  5 = **101**
  29 = **11101**
  and 21 =10**101**

After insertion of 11* we get the scenario as follows,

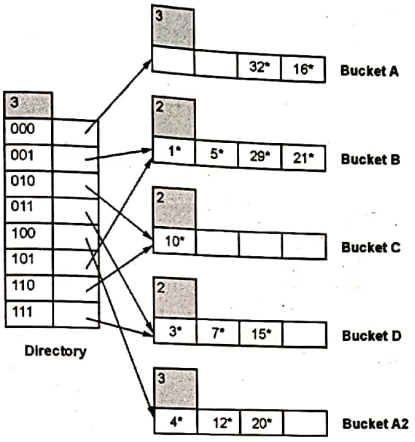


Fig. 6.5.4 : After insertion of 11*

Example 6.4.2 *The following key values are organized in an extendible hashing technique. 1 3 5 8 9 12 17 28. Show the extendible hash structure for this file if the hash function is h(x) = x mod 8 and buckets can hold three records. Show how extendable hash structure changes as the result of each of the following steps :*
Insert 2, Insert 24, Delete 5, Delete 12

Solution :

Step 1 : Initially we assume the hash function based on last two bits, of result of hash function.

1 mod 8 = 1 = 001

3 mod 8 = 3 = 011

5 mod 8 = 5 = 101

8 mod 8 = 0 = 000

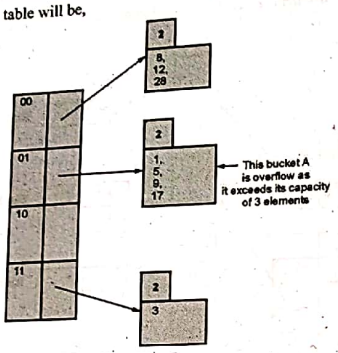
9 mod 8 = 1 = 001

12 mod 8 = 4 = 100

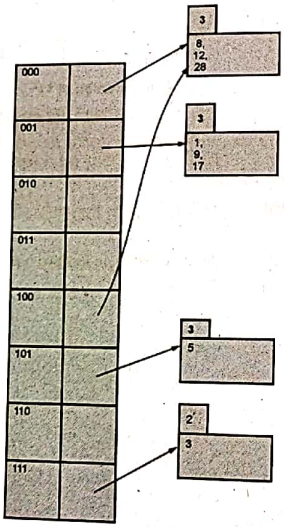
17 mod 8 = 1 = 001

28 mod 8 = 4 = 100

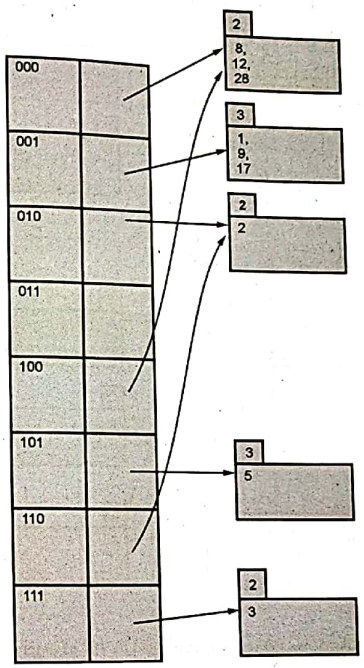
The extendible hash table will be,



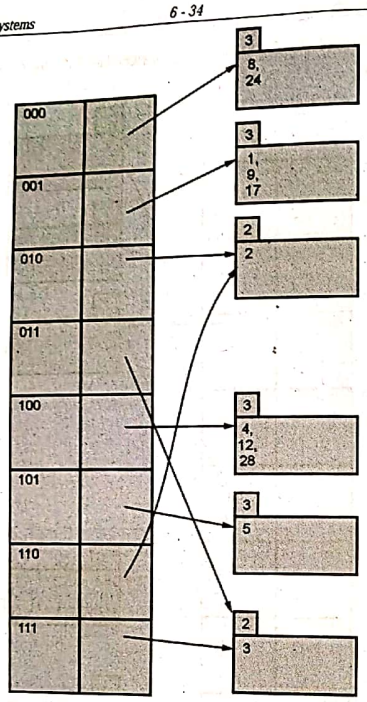
Hence we will extend the table by assuming the bit size as 3. The above indicated bucket A will split based on 001 and 101.



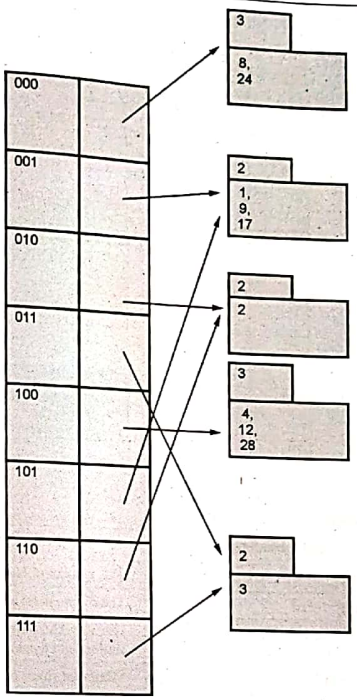
a) **Insert 2** will be 2 mod 8 = 2 = 010. If we consider last two digits i.e. 10 then there is no bucket. So we get,



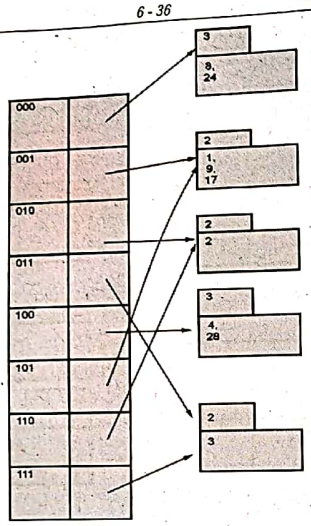
b) **Insert 24 :** 24 mod 8 = 0 = 000. The bucket in which 24 can be inserted is 8, 12, 28. But as this bucket is full we split it in two buckets based on digits 000 100.



c) **Delete 5 :** On deleting 5, we get one bucket pointed by 101 as empty. This will also result in reducing the local depth of the bucket pointed by 001. Hence we get,



(d) **Delete 12 :** We will simply delete 12 from the corresponding bucket there can not be any merging of buckets on deletion. The result of deletion is as given below -



Difference between Static and Dynamic Hashing

Sr. No.	Static Hashing	Dynamic Hashing
1.	The number of buckets are **fixed.**	The number of buckets are **not fixed.**
2.	**Chaining** is used	There is **no need of chaining.**
3.	Open hashing and closed hashing are forms of static hashing.	Extendible hashing and linear hashing are forms of dynamic hashing.
4.	Space overhead is more.	**Minimum space** overhead due to dynamic nature.
5.	As **file grows** the **performance** of static hash function **decreases.**	There is **no degradation in performance** when the file grows.
6.	The bucket address table is not required.	The **bucket address table** is required.
7.	The **bucket is directly accessed.**	The bucket address table is used to access the bucket.

Review Questions

1. *What is hashing? Explain static hashing and dynamic hashing with an example.*
2. *Explain various hashing techniques.*

6.6 Oral Questions and Answers

Q.1 What is index ?

Ans. : An index is a data structure that organizes data records on the disk to make the retrieval of data efficient.

Q.2 What are ordered indices ?

Ans. : This is type of indexing which is based on sorted ordering values. Various ordered indices are primary indexing, secondary indexing.

Q.3 What are the two types of ordered indices ?

Ans. : Two types of ordered indices are - Primary indexing and secondary indexing. The primary indexing can be further classified into dense indexing and sparse indexing and single level indexing and multilevel indexing.

Q.4 Give the comparison between ordered indices and hashing

Ans. : (1) If range of queries are common, ordered indices are to be used.

(2) The buckets containing records can be chained in sorted order in case of ordered indices.

(3) Hashing is generally better at retrieving records having a specified value of the key.

(4) Hash function assigns values randomly to buckets. Thus, there is no simple notion of "next bucket in sorted order."

Q.5 What are the causes of bucket overflow in a hash file organization ?

Ans. : Bucket overflow can occur for following reasons -

(1) Insufficient buckets : For the total number of buckets there are insufficient number of buckets to occupy.

(2) Skew : Some buckets are assigned more records than are others, so a bucket might overflow even while other buckets still have space. This situation is known as bucket skew.

Q.6 What can be done to reduce the occurrences of bucket overflows in a hash file organization ?

Ans. : (1) A bucket is a unit of storage containing one or more records (a bucket is typically a disk block).

(2) The file blocks are divided into M equal-sized buckets, numbered bucket0, bucket1... bucketM-1. Typically, a bucket corresponds to one (or a fixed number of) disk block.

(3) In a hash file organization we obtain the bucket of a record directly from its search-key value using a hash function, h (K).

(4) To reduce overflow records, a hash file is typically kept 70-80 % full.

(5) The hash function h should distribute the records uniformly among the buckets; otherwise, search time will be increased because many overflow records will exist.

Q.7 When is it preferable to use a dense index rather than a sparse index ? Explain your answer.

Ans. : 1. It is preferable to use a dense index instead of a sparse index when the file is not sorted on the indexed field.

2. Or when the index file is small compared to the size of memory.

Q.8 What are the disadvantages of B tree over B+ tree ?

Ans. : (1) Searching of a key value becomes difficult in B-tree as data can not be found in the leaf node.

(2) The leaf node can not store linked list and thus wastes the space.

Q.9 Mention different hashing techniques.

Ans. : Two types of hashing techniques are –

i) Static hashing ii) Dynamic hashing.

Q.10 List the mechanisms to avoid collision during hashing

Ans. : Collision Resolution techniques are :

(1) Separate chaining

(2) Open addressing techniques :

(i) Linear probing (ii) Quadratic probing

❑❑❑

7 Transaction Processing

Syllabus

Transaction processing *: Concurrency control, ACID property, Serializability of scheduling, Locking and timestamp based schedulers, Multi-version and optimistic Concurrency Control schemes, Database recovery.*

Contents

7.1 Basics of Transaction Processing

GTU : June-05, 08, Marks 4

Definition : A transaction can be defined as a **group of tasks** that form a single logical unit.

For example - Suppose we want to withdraw ₹ 100 from an account then we will follow following operations :

1) Check account balance
2) If sufficient balance is present request for withdrawal.
3) Get the money
4) Calculate Balance = Balance – 100
5) Update account with new balance.

The above mentioned **four steps** denote **one transaction.**

In a database, each transaction should maintain ACID property to meet the consistency and integrity of the database.

Review Question

1. *Explain transaction.* GTU : June 05, Mark 1, June 08, Marks 4

7.2 ACID Property

GTU : Dec.-05, 06, Summer-18, Winter-12,18, Marks 8

1) Atomicity :

- This property states that each transaction must be considered as a **single unit** and **must** be **completed fully or not completed at all.**
- No transaction in the database is left half completed.
- Database should be in a state either before the transaction execution or after the transaction execution. It **should not be in a state 'executing'.**
- For example - In above mentioned withdrawal of money transaction all the five steps must be completed fully or none of the step is completed. Suppose if transaction gets failed after step 3, then the customer will get the money but the balance will not be updated accordingly. The state of database should be either at before ATM withdrawal (i.e customer without withdrawn money) or after ATM withdrawal (i.e. customer with money and account updated). This will make the system in consistent state.

2) Consistency :

- The database must remain in consistent state after performing any transaction.
- For example : In ATM withdrawal operation, the balance must be updated appropriately after performing transaction. Thus the database can be in consistent state.

3) Isolation :

- In a database system where **more than one transaction** are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system.
- No transaction will affect the existence of any other transaction.
- **For example :** If a bank manager is checking the account balance of particular customer, then manager should see the balance either before withdrawing the money or after withdrawing the money. This will make sure that each individual transaction is completed and any other dependent transaction will get the consistent data out of it. Any failure to any transaction will not affect other transaction in this case. Hence it makes all the transactions consistent.

4) Durability :

- The database should be **strong enough** to handle any **system failure.**
- If there is any set of insert/update, then it should be able to handle and commit to the database.
- If there is any failure, the database should be able to recover it to the consistent state.
- For example : In ATM withdrawal example, if the system failure happens after Customer getting the money then the system should be strong enough to update Database with his new balance, after system recovers. For that purpose the system has to keep the log of each transaction and its failure. So when the system recovers, it should be able to know when a system has failed and if there is any pending transaction, then it should be updated to Database.

Review Questions

1. *What is transaction? Explain its properties.* GTU : Dec.-05, Marks 3
2. *What is transaction? Explain the ACID properties.* GTU : Winter-12, Marks 7
3. *List and Discuss ACID properties of transaction* GTU : Dec.- 06, Marks 8, Summer-18,, Winter-18, Marks 4

7.3 Transaction States

GTU : Dec.-06, Summer-18, Winter-14,15,18, Marks 4

- Each transaction has following five states :

1) **Active :** This is the first state of transaction. For example : insertion, deletion or updation of record is done here. But data is not saved to database.
2) **Partially Committed :** When a transaction executes its final operation, it is said to be in a partially committed state.

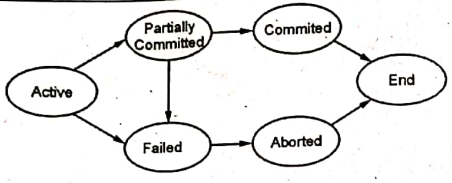


Fig. 7.3.1 Transaction states

3) **Failed :** A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

4) **Aborted :** If a transaction is failed to execute, then the database recovery system will make sure that the database is in its previous consistent state. If not, it brings the database to consistent state by aborting or rolling back the transaction.

5) **Committed :** If a transaction executes all its operations successfully, it is said to be committed. This is the last step of a transaction, if it executes without fail.

Example 7.3.1 *Define a transaction. Then discuss the following with relevant examples : 1. A read only transaction 2. A read write transaction 3. An aborted transaction*

Solution : (1) Read only transaction

T_1
Read(A)
Read(B)
Display(A-B)

(2) A read write transaction

T_1
Read(A)
A=A+100
Write(A)

(3) An aborted transaction

T1	T2	
Read(A)		Assume A=100
A=A+50		A=150
Write(A)		
	Read(A)	A=150
	A=A+100	A=250
RollBack		A=100 (restore back to original value which is before Transaction T1)
	Write(A)	

Review Questions

1. *Explain transaction state.* **GTU : Dec.-06, Marks 3**
2. *What is transaction ? Explain the properties of the transaction. Explain the states of the transaction with a neat sketch.* **GTU : Winter-14, Marks 4**
3. *During its execution, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. Explain why each state transition may occur.* **GTU : Winter-15, Marks 4**
4. *Define transaction. Explain various states of transaction with suitable diagram.* **GTU : Summer-18, Winter-18, Marks 4**

7.4 Schedules

GTU : Dec.-05,06, June-08, Winter-18, Summer-14,17,18, Marks 7

Schedule is an order of multiple transactions executing in concurrent environment. Following Fig. 7.4.1 represents the types of schedules.

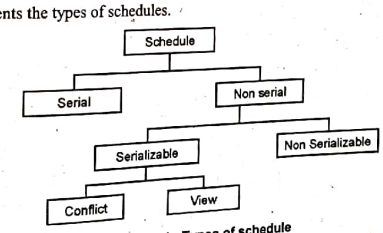


Fig. 7.4.1 : Types of schedule

Serial schedule : The schedule in which the transactions execute one after the other is called serial schedule. It is consistent in nature. For example : Consider following two transactions T_1 and T_2.

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

All the operations of transaction T_1 on data items A and then B executes and then in transaction T_2 all the operations on data items A and B execute. The **R stands for read** operation and W **stands for write** operation.

Non serial schedule : The schedule in which operations present within the transaction are intermixed. This may lead to conflicts in the result or inconsistency in the resultant data. For example -

Consider following two transactions,

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(B)
R(A)	
W(B)	
	R(B)
	W(B)

The above transaction is said to be non serial which result in inconsistency or conflicts in the data.

7.5 Serializability of Scheduling

GTU : Dec.-05,06,09, June-08, Summer-14, Marks 7

- When multiple transactions run concurrently, then it may lead to inconsistency of data (i.e. change in the resultant value of data from different transactions).
- Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.
- For example :

T_1	A	B	T_2
Initial Value	100	100	
A=A – 10			
W(A)			
B=B+10			
W(B)			
	90	110	
			A=A–10
			W(A)
	80	110	

- In above transactions initially T_1 will read the values from database as A=100, B=100 and modify the values of A and B. But transaction T_2 will read the modified value i.e. 90 and will modify it to 80 and perform write operation. Thus at the end of transaction T_1 value of A will be 90 but at end of transaction T_2 value of A will be 80. Thus conflicts or inconsistency occurs here. This sequence can be converted to a sequence which may give us consistent result. This process is called **serializability**.

Difference between Serial schedule and Serializable schedule

Serial schedule	Serializable schedule
No concurrency is allowed in serial schedule.	**Concurrency** is allowed in serializable schedule.
In serial schedule, if there are two transactions executing at the same time and no interleaving of operations is permitted, then following can be the possibilities of execution – (i) Execute all the operations of transactions T_1 in a sequence and then execute all the operations of transactions T_2 in a sequence. (ii) Execute all the operations of transactions T_2 in a sequence and then execute all the operations of transactions T_1 in a sequence.	In serializable schedule, if there are two transactions executing at the same time and interleaving of operations is allowed there can be different possible orders of executing an individual operation of the transactions.

Example of Serial schedule

T_1	T_2
Read(A)	
A=A–50	
Write(A)	
Read(B)	
B=B+100	
Write(B)	
	Read(A)
	A=A+10
	Write(A)

Example of Serializable schedule

T_1	T_2
Read(A)	
A=A–50	
Write(A)	
	Read(B)
	B=B+100
	Write(B)
Read(B)	
Write(B)	

- There are **two types of serializabilities :** Conflict serializability and view serializability

7.5.1 Conflict Serializability

Definition : Suppose T_1 and T_2 are two transactions and I_1 and I_2 are the instructions in T_1 and T_2 respectively. Then these two transactions are said to be conflict Serializable, if both the instruction access the data item d, and at least one of the instruction is write operation.

What is conflict ? : In the definition **three conditions** are specified for a conflict in conflict serializability -

1) There should be **different transactions**
2) The **operations** must be performed on **same data** items
3) **One of the operation** must be the **Write (W)** operation

- We can test a given schedule for conflict serializability by constructing a **precedence graph** for the schedule, and by searching for absence of cycles in the graph.
- Predence graph is a directed graph, consisting of G=(V,E) where V is set of vertices and E is set of edges. The set of vertices consists of all the transactions participating in the schedule. The set of edges consists of all edges $T_i \rightarrow T_j$ for which one of three conditions holds :
  1. T_i executes write(Q) before T_j executes read(Q).
  2. T_i executes read(Q) before T_j executes write(Q).
  3. T_i executes write(Q) before T_j executes write(Q).

- A serializability order of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called **topological sorting.**

Testing for serializability

Following method is used for testing the serializability : To test the conflict serializability we can draw a graph G=(V,E) where V = vertices which represent the number of transactions. E = edges for conflicting pairs.

Step 1 : Create a node for each transaction.

Step 2 : Find the conflicting pairs (RW, WR, WW) on the same variable (or data item) by different transactions.

Step 3 : Draw edge for the given schedule. Consider following cases

1. T_i executes write(Q) before T_j executes read(Q), then draw edge from T_i to T_j.
2. T_i executes read(Q) before T_j executes write(Q) , then draw edge from T_i to T_j
3. T_i executes write(Q) before T_j executes write(Q), , then draw edge from T_i to T_j

Step 4 : Now, if precedence graph is cyclic then it is a non conflict serializable schedule and if the precedence graph is acyclic then it is conflict serializable schedule.

Example 7.5.1 *Consider the following two transactions and schedule (time goes from top to bottom). Is this schedule conflict-serializable ? Explain why or why not.*

T_1	T_2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	

Solution :

Step 1 : To check whether the schedule is conflict serializable or not we will check from top to bottom. Thus we will start reading from top to bottom as

T_1: R(A) -> T_1:W(A) ->T_2:R(A) -> T_2:R(B) ->T_1:R(B) ->T_1:W(B)

Step 2 : We will find **conflicting operations.** Two operations are called as conflicting operations if all the following conditions hold true for them -

i) Both the **operations belong to different transactions.**

ii) Both the operations are on **same data item.**

iii) **At least one** of the two operations is a **write operation**

From above given example in the top to bottom scanning we find the conflict as T_1: W(A) ->T_2:R(A).

i) Here note that there are two different transactions T_1 and T_2,

ii) Both work on same data item i.e. A and

iii) One of the operation is write operation

Step 3 : We will build a precedence graph by drawing one node from each transaction. In above given scenario as there are two transactions, there will be two nodes namely T_1 and T_2

Step 4 : Draw the edge between conflicting transactions. For example in above given scenario, the conflict occurs while moving from T_1:W(A) to T_2:R(A). Hence edge must be from T_1 to T_2.

Step 5 : Repeat the step 4 while reading from top to bottom. Finally the **precedence graph** will be as follows

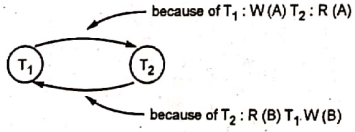


Fig. 7.5.1 : Precedence graph

Step 6 : Check if **any cycle** exists in the graph. Cycle is a path using which we can start from one node and reach to the same node. If the is cycle found then schedule is not conflict serializable. In the step 5 we get **a graph with cycle**, that means given schedule is **not conflict serializable.**

Example 7.5.2 *Check whether following schedule is conflict serializable or not. If it is not conflict serializable then find the serializability order.*

T_1	T_2	T_3
R(A)		
	R(B)	
		R(B)
	W(B)	
W(A)		
		W(A)
	R(A)	
	W(A)	

Solution :

Step 1 : We will read from top to bottom, and build a precedence graph for conflicting entries :

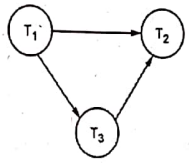


Fig. 7.5.2 Precedence graph

Step 2 : As there is **no cycle** in the precedence graph, the given sequence is **conflict serializable.** Hence we can convert this non serial schedule to serial schedule. For that purpose we will follow these steps to find the serializable order.

Step 3 : A **serializability order** of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called **topological sorting.**

Step 4 : Find the vertex which has no incoming edge which is T_1. Finally find the vertex having no outgoing edge which is T_2. So in between them is T_3. Hence the order will be $T_1 - T_3 - T_2$.

7.5.2 View Serializability

- If a given schedule is found to be view equivalent to some serial schedule, then it is called as a view serializable schedule.
- **View Equivalent Schedule :** Consider two schedules S_1 and S_2 consisting of transactions T_1 and T_2 respectively, then schedules S_1 and S_2 are said to be view equivalent schedule if it satisfies following three conditions :
  - If transaction T_1 reads a data item A from the database initially in schedule S_2, then in schedule S_2 also, T_1 must perform the initial read of the data item X from the database. This is same for all the data items. In other words - the initial reads must be same for all data items.
  - If data item A has been updated at last by transaction T_i in schedule S_1, then in schedule S_2 also, the data item A must be updated at last by transaction T_i.
  - If transaction T_i reads a data item that has been updated by the transaction T_j in schedule S_1, then in schedule S_2 also, transaction T_i must read the same data item that has been updated by transaction T_j. In other words the Write-Read sequence must be same.

Steps to check whether the given schedule is view serializable or not

Step 1 : If the schedule is conflict serializable then it is surely view serializable because conflict serializability is a restricted form of view serializability.

Step 2 : If it is not conflict serializable schedule then check whether there exist any blind write operation. The blind write operation is a write operation without reading a value. If there does not exist any blind write then that means the given schedule is not view serializable. In other words if a blind write exists then that means schedule may or may not be view conflict.

Step 3 : Find the view equivalence schedule

Example 7.5.3 *Consider the following schedules for checking if these are view serializable or not.*

T_1	T_2	T_3
		W(C)
	R(A)	
	W(B)	R(B)
R(C)		
		W(B)
W(B)		

Solution :

i) The initial read operation is performed by T_2 on data item A or by T_1 on data item C. Hence we will begin with T_2 or T_1. We will choose T_2 at the beginning.

ii) The final write is performed by T_1 on the same data item B. Hence T_1 will be at the last position.

iii) The data item C is written by T_3 and then it is read by T_1. Hence T_3 should appear before T_1.

Thus we get the order of schedule of view serializability as $T_2 - T_1 - T_3$

Example 7.5.4 *Consider following two transactions :*

T_1 : read(A)
read(B)
if A=0 then B:=B+1;
write(B)

T_2 : read(B);
read(A);
if B=0 then A:=A+1;
write(A)

Let consistency requirement be A=0 V B=0 with A=B=0 the initial values.

1) *Show that every serial execution involving these two transactions preserves the consistency of the Database ?*
2) *Show a concurrent execution of T_1 and T_2 that produces a non serializable schedule ?*
3) *Is there a concurrent execution of T_1 and T_2 that produces a serializable schedule ?*

Solution : 1) There are two possible executions : **T_1 -> T_2 or T_2->T_1**

Consider case T_1->T_2 then

A	B
0	0
0	1
0	1

A∨B =A OR B=F∨T=T. This means consistency is met.

Consider case T_2->T_1 then

A	B
0	0
1	0
1	0

A∨B = A OR B = F∨T = T. This means consistency is met.

(2) The concurrent execution means interleaving of transactions T_1 and T_2. It can be

T_1	T_2
R(A)	
	R(B)
	R(A)
R(B) If A=0 then B=B+1	If B=0 then A=A+1 W(A)
W(B)	

This is a non-serializable schedule.

(3) There is no concurrent execution resulting in a serializable schedule.

Example 7.5.5 *Test serializability of the following schedule :*
i) $r_1(x);r_3(x);w_1(x);r_2(x);w_3(x)$ ii) $r_3(x);r_2(x);w_3(x);r_1(x);w_1(x)$

Solution : i) $r_1(x);r_3(x);w_1(x);r_2(x);w_3(x)$

The r_1 represents the read operation of transaction T_1, w_3 represents the write operation on transaction T_3 and so on. Hence from given sequence the schedule for three transactions can be represented as follows :

T_1	T_2	T_3
$r_1(x)$		
		$r_3(x)$
$w_1(x)$		
	$r_2(x)$	
		$w_3(x)$

Step 1 : We will use the precedence graph method to check the serializability. As there are three transactions, three nodes are created for each transaction.

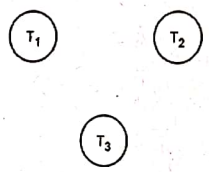


Fig. 7.5.3 Nodes

Step 2 : We will read from top to bottom. Initially we read $r_1(x)$ and keep on moving bottom in search of write operation. Here all the transactions work on same data item i.e. x. Now we get a write operation in T_3 as $w_3(x)$. Hence the dependency is from T_1 to T_3. Therefore we draw edge from T_1 to T_3.

Similarly, for $r_3(x)$ we get $w_1(x)$ pair. Hence there will be edge from T_3 to T_1. Continuing in this fashion we get the precedence graph as

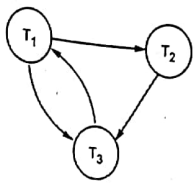


Fig. 7.5.4 Precedence Graph

Step 3 : As cycle exists in the above precedence graph, we conclude that it is **not serializable.**

ii) $r_3(x);r_2(x);w_3(x);r_1(x);w_1(x)$

From the given sequence the schedule can be represented as follows :

T_1	T_2	T_3
		$r_3(x)$
	$r_2(x)$	
		$w_3(x)$
$r_1(x)$		
$w_1(x)$		

Step 1 : Read the schedule from top to bottom for pair of operations. For $r_3(x)$ we get $w_1(x)$ air. Hence edge exists from T_3 to T_1 in precedence graph.

There is a pair from $r_2(x) : w_3(x)$. Hence edge exists from T_2 to T_3.

There is a pair from $r_2(x) : w_1(x)$. Hence edge exists from T_2 to T_1.

There is a pair from $w_3(x): r_1(x)$. Hence edge exists from T_3 to T_1.

Step 2 : The precedence graph will then be as follows -

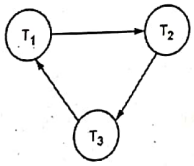
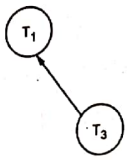


Fig. 7.5.5 Precedence graph

Step 3 : As there is no cycle in the above graph, the given schedule is **serializable.**

Step 4 : The searializability order for consistent schedule will be obtained by applying topological sorting on above drawn precedence graph. This can be achieved as follows,

Sub-Step 1 : Find the node having no incoming edge. We obtain T_2 is such a node. Hence T_2 is at the beginning of the serializability sequence. Now delete T_2. The Graph will be



Sub-Step 2 : Repeat sub-Step 1, We obtain T_3 and T_1 nodes as a sequence.

Thus we obtain the sequence of transactions as T_2, T_3 and T_1. Hence the serializability order is $r_2(x);r_3(x);w_3(x):r_1(x);w_1(x)$

Example 7.5.6 *Consider the following schedules. The actions are listed in the order they are scheduled, and prefixed with the transaction name.*

S_1 : T_1 : R(X), T_2 : R(X), T_1 : W(Y), T_2 : W(Y) T_1 : R(Y), T_2 : R(Y)

S_2 : T_3 : W(X), T_1 : R(X), T_1 : W(Y), T_2 : R(Z). T_2 : W(Z) T_3 : R(Z)

For each of the schedules, answer the following questions :

i) What is the precedence graph for the schedule ?

ii) Is the schedule conflict-serializable ? If so, what are all the conflict equivalent serial schedules ?

iii) Is the schedule view-serializable ? If so, what are all the view equivalent serial schedules ?

Solution : i) We will find **conflicting operations**. Two operations are called as conflicting operations if all the following conditions hold true for them -

- Both the **operations belong to different transactions.**
- Both the operations are on **same data item.**
- At least one of the two operations is a write operation

For S_1 : From above given example in the top to bottom scanning we find the conflict as

- T_1 : W(Y), T_2 : W(Y) and
- T_2 : W(Y), T_1 : R(Y)

Hence we will build the precedence graph. Draw the edge between conflicting transactions. For example in above given scenario, the conflict occurs while moving from T_1: W(Y) to T_2:W(Y). Hence edge must be from T_1 to T_2. Similarly for second conflict, there will be the edge from T_2 to T_1.

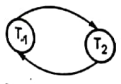


Fig. 7.5.6 Precedence graph for S_1

For S_2 : The conflicts are

- T_3 : W(X), T_1 : R(X)
- T_2 : W(Z) T_3 : R(Z)

Hence the precedence graph is as follows -

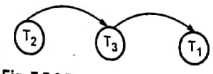


Fig. 7.5.6 Precedence graph for S_2

(ii)

- S_1 is **not conflict-serializable** since the dependency **graph has a cycle.**
- S_2 is conflict-serializable as the dependency graph is acylic. The order T_2-T_3-T_1 is the only equivalent **serial order.**

(iii)

- S_1 is not view serializable.
- S_2 is trivially view-serializable as it is conflict serializable. The only serial order allowed is T_2-T_3-T_1.

Review Questions

1. *Explain distinction between serial schedule and searializable schedule.* **GTU : Dec.-05, Marks 3**
2. *Explain conflict serializability.* **GTU : Dec.-06, Marks 3**
3. *Compare view serializability and conflict serializability* **GTU : June-08, Marks 6; Summer-14, Marks 7**
4. *What is serializability ? What is its objective ?* **GTU : Summer-17, Marks 1**
5. *Write a note on conflict serializability.* **GTU : Summer-18, Winter-18, Marks 7**

7.6 Concurrency Control

GTU : Dec.-09, May-11, Winter-15, Marks 7

- One of the fundamental properties of a transaction is **isolation.**
- When several transactions execute concurrently in the database, however, the isolation property may no longer be preserved.
- A database can have multiple transactions running at the same time. This is called **concurrency.**

- To preserve the isolation property, the system must control the interaction among the concurrent transactions; this control is achieved through one of a variety of mechanisms called **concurrency control schemes.**
- **Definition of concurrency control :** A mechanism which ensures that simultaneous execution of more than one transactions does not lead to any database inconsistencies is called **concurrency control mechanism.**
- The concurrency control can be achieved with the help of various protocols such as - lock based protocol, Deadlock handling, Multiple Granularity, Timestamp based protocol, and validation based protocols.

7.6.1 Need for Concurrency

Following are the purposes of concurrency control –

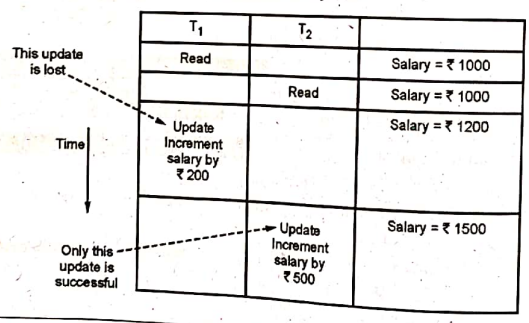
- **To ensure isolation**
- **To resolve read-write or write-write conflicts**
- **To preserve consistency of database**

- Concurrent execution of transactions over shared database creates several data integrity and consistency problems – these are

(1) Lost Update Problem : This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

For example – Consider following transactions

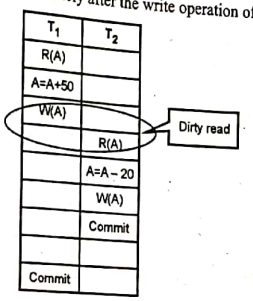
(1) Salary of Employee is read during transaction T_1.
(2) Salary of Employee is read by another transaction T_2.
(3) During transaction T_1, the salary is incremented by ₹ 200
(4) During transaction T_2, the salary is incremented by ₹ 500



T_1	T_2	
Read		Salary = ₹ 1000
	Read	Salary = ₹ 1000
Update Increment salary by ₹ 200		Salary = ₹ 1200
	Update Increment salary by ₹ 500	Salary = ₹ 1500

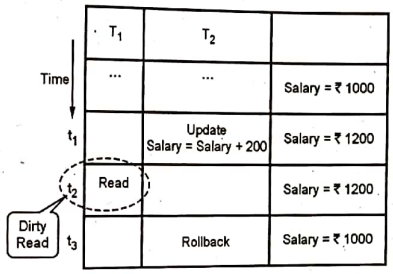
The result of the above sequence is that the update made by transaction T1 is completely lost. Therefor this problem is called as lost update problem.

(2) Dirty Read or Uncommited Read problem : The dirty read is a situation in which one transaction reads the data immediately after the write operation of previous transaction



T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A – 20
	W(A)
	Commit
Commit	

For example - Consider following transactions -
Assume initially salary is = ₹ 1000



	T_1	T_2	
	...	...	Salary = ₹ 1000
t_1		Update Salary = Salary + 200	Salary = ₹ 1200
t_2	Read		Salary = ₹ 1200
t_3		Rollback	Salary = ₹ 1000

(1) At the time t_1, the transaction T_2 updates the salary to ₹1200
(2) This salary is read at time t_2 by transaction T_1. Obviously it is ₹ 1200
(3) But at the time t_3, the transaction T_2 performs Rollback by undoing the changes made by T_1 and T_2 at time t_1 and t_2.
(4) Thus the salary again becomes = ₹ 1000. This situation leads to **Dirty Read or Uncommited Read** because here the read made at time t_2**(immediately after update of another transaction)** becomes a dirty read.

(3) Non-repeatable read problem : This problem is also known as **inconsistent analysis problem.** This problem occurs when a particular transaction sees two different values for the same row within its lifetime. For example -

Time	T_1	T_2	
t_1	Read		Salary = ₹ 1000
t_2		Update salary from ₹ 1000 to ₹ 1200	Salary = ₹ 1200
t_3		Commit	
t_4	Read		Salary = ₹ 1200

(1) At time t_1, the transaction T_1 reads the salary as ₹ 1000

(2) At time t_2 the transaction T_2 reads the same salary as ₹ 1000 and updates it to ₹1200

(3) Then at time t_3, the transaction T_2 gets committed.

(4) Now when the transaction T_1 reads the same salary at time t_4, it gets different value than what it had read at time t_1. Now, transaction T_1 cannot repeat its reading operation. Thus inconsistent values are obtained.

Hence the name of this problem is non-repeatable read or inconsistent analysis problem.

(4) Phantom read problem

The phantom read problem is a special case of non repeatable read problem.

This is a problem in which one of the transaction makes the changes in the database system and due to these changes another transaction can not read the data item which it has read just recently. For example –

Time	T_1	T_2	
t_1	Read		Salary = ₹ 1000
t_2		Read	Salary = ₹ 1000
t_3	Delete salary		No salary
t_4		Read	"Can not find salary"

(1) At time t_1, the transaction T_1 reads the value of salary as ₹ 1000

(2) At time t_2, the transaction T_2 reads the value of the same salary as ₹ 1000

(3) At time t_3, the transaction T_1 deletes the variable salary.

(4) Now at time t_4, when T_2 again reads the salary it gets error. Now transaction T_2 can not identify the reason why it is not getting the salary value which is read just few time back.

This problem occurs due to changes in the database and is called phantom read problem.

Example 7.6.1 *Justify the following statement : "Concurrent execution of transactions is more important when data must be fetched from (slow) disk or when transactions are long, and is less important when data is in memory and transactions are very short."*

Solution :

(1) If transaction is very long or when it fetches data from slow disk, it takes a long time to complete.

(2) In absence of concurrency, other transactions will have to wait for longer period of time. Average response time will increase.

(3) Also when the transaction is reading data from disk, CPU is idle. So resources are not properly utilized. Hence concurrent execution becomes important in this case.

(4) However, when the transactions are short or the data is available in memory, these problems do not occur.

Review Questions

1. *Why concurrency control is needed ?* **GTU : Dec.09, Marks 3**
2. *What is concurrency ? What are the three problems due to concurrency ? How the problems can be avoided, explain for one of the three problems.* **GTU : May 11, Marks 7**
3. *Explain the dirty read problem.* **GTU : Winter-15, Marks 3**

7.7 Locking Schedulers

GTU : Winter-12,15,17,18, Summer-13,17,18, Marks 7

7.7.1 Why Do We Need Locks ?

- One of the method to ensure the **isolation** property in transactions is to require that data items be accessed in a mutually exclusive manner. That means, while one transaction is accessing a data item, no other transaction can modify that data item.
- The most common method used to implement this requirement is to allow a transaction to access a data item only if it is currently holding a **lock on that item.**
- Thus the lock on the operation is required to ensure the isolation of transaction.

7.7.2 Simple Lock Based Protocol

- **Concept of protocol :** The lock based protocol is a mechanism in which there is exclusive use of **locks** on the data item for current transaction.
- **Types of locks :** There are two types of locks used –

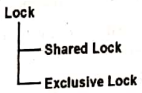


Fig. 7.7.1 Types of locks

i) **Shared lock :** The shared lock is used for reading data items only. It is denoted by Lock-S. This is also called as read lock.

ii) **Exclusive lock :** The exclusive lock is used for both read and write operations. It is denoted as Lock-X. This is also called as write lock.

- The **compatibility matrix** is used while working on set of locks. The **concurrency control manager** checks the compatibility matrix before granting the lock. If the two modes of transactions are compatible to each other then only the lock will be granted.
- In a set of locks may consists of shared or exclusive locks. Following matrix represents the compatibility between modes of locks.

	S	X
S	T	F
X	F	F

Fig. 7.7.2 Compatibility matrix for locks

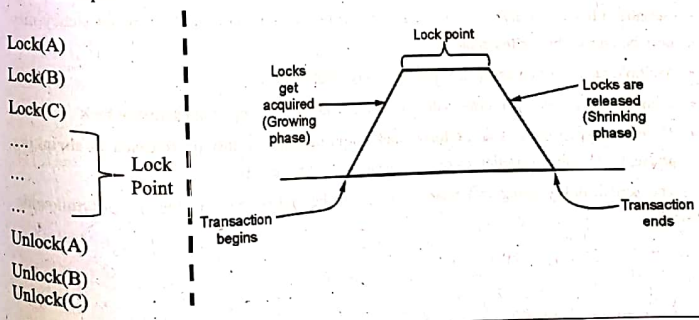
Here T stands for True and F stands for False. If the control manager get the compatibility mode as True then it grant the lock otherwise the lock will be denied.

- **For example :** If the transaction T_1 is holding a shared lock in data item A, then the control manager can grant the shared lock to transaction T_2 as compatibility is True. But it cannot grant the exclusive lock as the compatibility is false. In simple words if transaction T_1 is reading a data item A then same data item A can be read by another transaction T_2 but cannot be written by another transaction.
- Similarly if an exclusive lock (i.e. lock for read and write operations) is hold on the data item in some transaction then no other transaction can acquire Shared or exclusive lock as the compatibility function denotes F. That means of some transaction is writing a data item A then another transaction can not read or write that data item A.
- Hence the **rule of thumb is**
  i) **Any number** of transactions can hold **shared lock** on an item.
  ii) But **exclusive lock can be hold by only one transaction.**
- **Example of a schedule denoting shared and exclusive locks :** Consider following schedule in which initially A=100. We deduct 50 from A in T_1 transaction and Read the data item A in transaction T_2. The scenario can be represented with the help of locks and concurrency control manager as follows :

	T_1	T_2	Concurrency control manager
Exclusive lock	Lock-X(A)		
			Grant X(A,T_1) because in T_1 there is write operation.
	R(A)		
	A=A – 50		
	W(A)		
	Unlock(A)		
		Lock-S(A)	
Shared lock			Grant S(A,T_2) because in T_2 there is Read operation
		R(A)	
		Unlock(A)	

7.7.3 Two Phase Locking

- The two phase locking is a protocol in which there are two phases :
  i) **Growing phase (Locking phase) :** It is a phase in which the transaction may obtain locks but does not release any lock.
  ii) **Shrinking phase (Unlocking phase) :** It is a phase in which the transaction may release the locks but does not obtain any new lock.
- **Lock Point** : The **last lock position** or **first unlock position** is called lock point.
- For example -



Consider following transactions

T_1	T_2
Lock-X(A)	Lock-S(B)
Read(A)	Read(B)
A=A–50	Unlock-S(B)
Write(A)	
Lock-X(B)	
Unlock-X(A)	
B=B+100	Lock-S(A)
Write(B)	Read(A)
Unlock-X(B)	Unlock-S(A)

The important rule for being a two phase locking is - **All Lock operations precede all the unlock operations.**

In above transactions T_1 is in two phase locking mode but transaction T_2 is not in two phase locking. Because in T_2, the Shared lock is acquired by data item B, then data item B is read and then the lock is released. Again the lock is acquired by data item A , then the data item A is read and the lock is then reloased. Thus we get lock-unlock-lock-unlock sequence. Clearly this is not possible in two phase locking.

Example 7.7.1 *Prove that two phase locking guarantees serializability.*

Solution :

- Serializability is mainly an issue of handling write operation. Because any inconsistency may only be created by **write** operation.
- Multiple reads on a database item can happen parallely.
- 2-Phase locking protocol restricts this unwanted read/write by applying **exclusive lock.**
- Moreover, when there is an **exclusive lock** on an item it will **only be released in shrinking phase.** Due to this restriction there is no chance of getting any inconsistent state.

The serializability using two phase locking can be understood with the help of following example

Consider two transactions

T_1	T_2
R(A)	
	R(A)
R(B)	
W(B)	

Step 1 : Now we will **apply two phase locking**. That means we will **apply locks in growing and shrinking phase**

T_1	T_2
Lock-S(A)	
R(A)	
	Lock-S(A)
	R(A)
Lock-X(B)	
R(B)	
W(B)	
Unlock-X(B)	
	Unlock-S(A)

Note that above schedule is serializable as it prevents interference between two transactions. The serializability order can be obtained based on the **lock point**. The lock point is either last lock operation position or first unlock position in the transaction.

The last lock position is in T_1, then it is in T_2. Hence the serializability will be $T_1 \rightarrow T_2$ based on lock points. Hence The **serializability sequence** can be $R_1(A);R_2(A);R_1(B);W_1(B)$

Advantages of two phase locking

(1) It ensures serializability.

Disadvantages of two phase locking protocol

(1) It leads to dealocks.

(2) It leads to cascading rollback.

Problems in two phase locking

The two phase locking protocol leads to two problems – deadlock and cascading roll back.

(1) Deadlock : The deadlock problem can not be solved by two phase locking. Deadlock is a situation in which when two or more transactions have got a lock and waiting for another locks currently held by one of the other transactions.

For example

T_1	T_2
Lock-X(A)	Lock-X(B)
Read(A)	Read(B)
A=A–50	B=B+100
Write(A)	Write(B)
Delayed, wait for T_2 to release Lock on B	Delayed, wait for T_1 to release Lock on A

(2) Cascading Rollback : Cascading rollback is a situation in which a single transaction failure leads to a series of transaction rollback. For example –

T_1	T_2	T_3
Read(A)		
Read(B)		
C=A+B		
Write(C)		
	Read(C)	
	Write(C)	
		Read(C)

When T_1 writes value of C then only T_2 can read it. And when T_2 writes the value of C then only transaction T_3 can read it. But if the transaction T_1 gets failed then automatically transactions T_2 and T_3 gets failed.

The simple two phase locking does not solve the cascading rollback problem. To solve the problem of cascading Rollback two types of two phase locking mechanisms can be used.

7.7.3.1 Types of Two Phase Locking

(1) Strict two phase locking : The strict 2PL protocol is a basic two phase protocol but **all the exclusive mode locks** be **held until the transaction commits**. That means in other words all the **exclusive locks** are **unlocked** only after the **transaction is committed**. That also means that if T_1 has exclusive lock, then T_1 will release the exclusive lock only after commit operation, then only other transaction is allowed to read or write. For example - Consider two transactions

T_1	T_2
W(A)	
	R(A)

If we apply the locks then

T_1	T_2
Lock-X(A)	
W(A)	
Commit	
Unlock(A)	
	Lock-S(A)
	R(A)
	Unlock-S(A)

Thus only after commit operation in T_1, we can unlock the exclusive lock. This ensures the strict serializability.

Thus compared to basic two phase locking protocol, the advantage of strict 2PL protocol is it ensures strict serializability.

(2) Rigorous two phase locking : This is stricter two phase locking protocol. Here all locks are to be held until the transaction commits. The transactions can be seriealized in the order in which they commit.

example - Consider transactions

T_1
R(A)
R(B)
W(B)

If we apply the locks then

T_1
Lock-S(A)
R(A)
Lock-X(B)
R(B)
W(B)
Commit
Unlock(A)
Unlock(B)

Thus the above transaction uses rigorous two phase locking mechanism

Example 7.7.2 *Consider the following two transactions :*
T_1:read(A)Read(B);
If A=0 then B=B+1;
Write(B)
T_2:read(B); read(A)
If B=0 then A=A+1
Write(A)
Add lock and unlock instructions to transactions T_1 and T_2 so that they observe two phase locking protocol. Can the execution of these transactions result in deadlock ?

Solution :

T_1	T_2
Lock-S(A)	Lock-S(B)
Read(A)	Read(B)
Lock-X(B)	Lock-X(A)
Read(B)	Read(A)
if A=0 then B=B+1	if B=0 then A=A+1
Write(B)	Write(A)
Unlock(A)	Unlock(B)
Commit	Commit
Unlock(B)	Unlock(A)

This is lock-unlock instruction sequence help to satisfy the requirements for strict two phase locking for the given transactions.

The execution of these transactions result in deadlock. Consider following partial execution scenario which leads to deadlock.

T_1	T_2
Lock-S(A)	Lock-S(B)
Read(A)	Read(B)
Lock-X(B)	Lock-X(A)
Now it will wait for T_2 to release exclusive lock on A	Now it will wait for T_1 to release exclusive lock on B

Review Questions

1. *Explain two phase locking with its advantages and disadvantages* **GTU : Winter-12, 17, Summer-13, Marks 7**
2. *Explain three concurrency problem. How does the strict two-phase locking protocol solve three problems of concurrency? Explain with example.* **GTU : Summer-17, Marks 7**
3. *Explain various types of locks used in lock based protocol for concurrency control* **GTU : Winter-17, Marks 3**

4. *Write differences between shared lock and exclusive lock.* **GTU : Summer-18, Winter-18, Marks 3**
5. *Write a note on two phase locking protocol.* **GTU : Summer-18, Winter-15, 18,Marks 4**
6. *Write a note on - two phase commit protocol.* **GTU : Summer-18, Winter-18, Marks 3**

7.8 Time Stamp based Scheduler

GTU : Dec.-06, Winter-14, Summer-14, Marks 7

- The time stamp ordering protocol is a scheme in which the order of transaction is decided in advance based on their timestamps. Thus the schedules are serialized according to their timestamps.
- The timestamp-ordering protocol ensures that any conflicting read and write operations are executed in timestamp order.
- A larger timestamp indicates a more recent transaction or it is also called as **younger transaction** while lesser timestamp indicates **older transaction**
- Assume a collection of data items that are accessed, with read and write operations, by transactions.
- For each data item X the DBMS maintains the following values: –
  - **RTS(X):** The Timestamp on which object X was last read (by some transaction T_i, i.e., RTS(X):=TS(T_i)) [Note that: RTS stands for Read Time Stamp]
  - **WTS(X):** The Timestamp on which object X was last written (by some transaction T_j, i.e., WTS(X):=TS(T_j)) [Note that: WTS stands for Write Time Stamp]
- For the following algorithms we use the following assumptions: – A data item X in the database has a **RTS(X) and WTS(X)**. These are actually the timestamps of Read and write operations performed on data item X at latest time.
- A transaction T attempts to perform some action (read or write) on data item X on some timestamp and we call that timestamp as **TS(T).**
- By timestamp ordering algorithm we need to decide whether transaction T has to be aborted or T can continue execution.

Basic Timestamp Ordering Algorithm

Case 1 (Read) : Transaction T issues a **read(X) operation**

i) If TS(T) < WTS(X), then read(X) is rejected. T has to abort and be rejected.

ii) If WTS(X) ≤ TS(T), then execute read(X) of T and update RTS(X).

Case 2 (Write) : Transaction T issues a **write(X) operation**

i) If TS(T) < RTS(X) or if TS(T) < WTS(X), then write is rejected

ii) If RTS(X) ≤ TS(T) or WTS(X) ≤ TS(T), then execute write(X) of T and update WTS(X).

Example for Case 1 (Read operation)

(i) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec and 20 sec respectively.

10 Sec T_1	20 Sec T_2
R(X)	
	W(X)
R(X)	

RTS(X) and WTS(X) is initially = 0

Then RTS(X)=10, when transaction T_1 executes

After that WTS(X) =20 when transaction T_2 executes

Now if Read operation R(X) occurs on transaction T_1 at TS(T_1)=10 then

TS(T_1) i.e. 10 <WTS(X) i.e. 20, hence we have to reject second read operation on T_1 i.e.

10 Sec T_1	20 Sec T_2
R(X)	
	W(X)
R(X)	

This read operation gets rejected as it occurs at older timestamp than the write operation

(ii) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec and 20 sec respectively.

10 Sec T_1	20 Sec T_2
W(X)	
	R(X)

RTS(X) and WTS(X) is initially = 0

Then WTS(X) =10 as transaction T_1 executes.

Now if Read operation R(X) occurs on transaction T_2 at TS(T_2)=20 then

$TS(T_2)$ i.e. 20 >WTS(X) which is 10, hence we accept read operation on T_2. The transaction T_2 will perform read operation and now RTS will be updated as

RTS(X)=20

Example for Case 2 (Write Operation)

(i) Suppose we have two transactions T1 and T2 with timestamps 10 sec and 20 sec respectively.

10 Sec T_1	20 Sec T_2
R(X)	
	W(X)
W(X)	

RTS(X) and WTS(X) is initially = 0

Then RTS(X)=10, when transaction T_1 executes

After that WTS(X) =20 when transaction T_2 executes

Now if Write operation W(X) occurs on transaction T_1 at $TS(T_1)$=10 then

$TS(T_1)$ i.e. 10 <WTS(X), hence we have to reject second write operation on T_1 i.e.

10 Sec T_1	20 Sec T_2
R(X)	
	W(X)
W(X)	

This write operation gets rejected as it occurs at older timestamp than the write operation at transaction T_2

(ii) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec and 20 sec respectively.

10 Sec T_1	20 Sec T_2
W(X)	
	W(X)

RTS(X) and WTS(X) is initially = 0

Then WTS(X) =10 as transaction T_1 executes.

Now if write operation W(X) occurs on transaction T_2 at $TS(T_2)$=20 then

$TS(T_2)$ i.e. 20 >WTS(X) which is 10, hence we accept write operation on T_2. The transaction T_2 will perform write operation and now WTS will be updated as

WTS(X)=20

Advantages and Disadvantages of time stamp ordering

Advantages

(1) Schedules are serializable

(2) No waiting for transaction and hence there is no deadlock situation.

Disadvantages

(1) Schedules are not recoverable once transactions occur.

(2) Same transaction may be continuously aborted or restarted.

Review Question

1. *Explain time-stamp based protocol.*

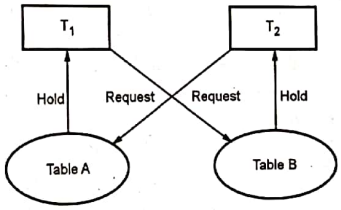
GTU : Dec 06, Marks 3, Winter-14, Summer-14, Marks 7

7.9 Deadlock

GTU : Dec.-10, Marks 7

Deadlock is a specific concurrency problem in which two transactions depend on each other for something.

For example - Consider that transaction T_1 holds a lock on some rows of table **A** and needs to update some rows in the **B** table. Simultaneously, transaction T_2 holds locks on some rows in the **B** table and needs to update the rows in the **A** table held by Transaction T_1.



Now, the main problem arises. Now Transaction T_1 is waiting for T_2 to release its lock and similarly, transaction T_2 is waiting for T_1 to release its lock. All activities come to a halt state and remain at a standstill. This situation is called **deadlock** in DBMS.

Definition : Deadlock can be formally defined as - " A system is in deadlock state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set. "

There are **four conditions for a deadlock to occur**

A deadlock may occur if all the following conditions holds true.

1. **Mutual exclusion condition :** There must be at least one resource that cannot be used by more than one process at a time.
2. **Hold and wait condition :** A process that is holding a resource can request for additional resources that are being held by other processes in the system.
3. **No preemption condition :** A resource cannot be forcibly taken from a process. Only the process can release a resource that is being held by it.
4. **Circular wait condition :** A condition where one process is waiting for a resource that is being held by second process and second process is waiting for third process ....so on and the last process is waiting for the first process. Thus making a circular chain of waiting.

Deadlock can be handled using two techniques –

1. Deadlock prevention 2. Deadlock detection and deadlock recovery

1. Deadlock prevention :

For large database, deadlock prevention method is suitable. A deadlock can be prevented if the resources are allocated in such a way that deadlock never occur. The DBMS analyzes the operations whether they can create deadlock situation or not, If they do, that transaction is never allowed to be executed.

There are two techniques used for deadlock prevention -

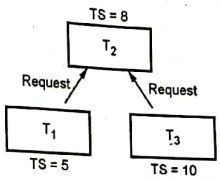
(i) Wait-Die :

- In this scheme, if a transaction requests for a resource which is already held with a conflicting lock by another transaction then the DBMS simply checks the timestamp of both transactions. It **allows the older transaction to wait** until the resource is available for execution.
- Suppose there are two transactions T_i and T_j and let TS(T) is a timestamp of any transaction T. If T_2 holds a lock by some other transaction and T_1 is requesting for resources held by T_2 then the following actions are performed by DBMS :
  - Check if $TS(T_i) < TS(T_j)$ - If T_i is the older transaction and T_j has held some resource, then T_i is allowed to wait until the data-item is available for execution. That means if the older transaction is waiting for a resource which is locked by the younger transaction, then the older transaction is allowed to wait for resource until it is available.
  - Check if $TS(T_i) < TS(T_j)$ - If T_i is older transaction and has held some resource and if T_j is waiting for it, then T_j is killed and restarted later with the random delay but with the same timestamp.

Timestamp is a way of assigning priorities to each transaction when it starts. If timestamp is lower then that transaction has higher priority. **That means oldest transaction has highest priority.**

For example -

Let T_1 is a transaction which requests the data item acquired by Transaction T_2. Similarly T_3 is a transaction which requests the data item acquired by transaction T_2.



Here $TS(T_1)$ i.e. Time stamp of T_1 is less than $TS(T_3)$. In other words T_1 is older than T_3. Hence T_1 is made to **wait** while T_3 is **rolledback.**

(ii) Wound- wait :

- In wound wait scheme, if the older transaction requests for a resource which is held by the younger transaction, then older transaction forces younger one to kill the transaction and release the resource. After some delay, the younger transaction is restarted but with the same timestamp.
- If the older transaction has held a resource which is requested by the Younger transaction, then the younger transaction is asked to wait until older releases it.

Suppose T_1 needs a resource held by T_2 and T_3 also needs the resource held by T_2, with $TS(T_1)=5$, $TS(T_2)=8$ and $TS(T_3)=10$, then T_1 being older waits and T_3 being younger dies. After the some delay, the younger transaction is restarted but with the same timestamp.

This ultimately prevents a deadlock to occur.

To summarize

	Wait-Die	Wound-wait
Older transaction needs a data item held by younger transaction	older transaction waits	younger transaction dies.
Younger transaction needs a data item held by older transaction	Younger transaction dies	Younger transaction dies.

2. Deadlock detection :

- In deadlock detection mechanism, an algorithm that examines the state of the system is invoked periodically to determine whether a deadlock has occurred or not. If deadlock is occurrence is detected, then the system must try to recover from it.
- Deadlock detection is done using wait for graph method.

Wait for graph

- In this method, a graph is created based on the transaction and their lock. If the created **graph has a cycle or closed loop, then there is a deadlock.**
- The wait for the graph is maintained by the system for every transaction which is waiting for some data held by the others. The system keeps checking the graph if there is any cycle in the graph.
- This graph consists of a pair G = (V, E), where V is a set of vertices and E is a set of edges.
- The set of vertices consists of all the transactions in the system.
- When transaction T_i requests a data item currently being held by transaction T_j , then the edge $T_i \rightarrow T_j$ is inserted in the wait-for graph. This edge is removed only when transaction T_j is no longer holding a data item needed by transaction T_i .

For example - Consider following transactions, We will draw a wait for graph for this scenario and check for deadlock.

T_1	T_2
R(A)	
	R(A)
W(A)	
R(B)	
	W(A)
W(B)	

We will use three rules for designing the wait-for graph-

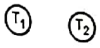
Rule 1 : If T_1 has **Read** operation and then T_2 has **Write** operation then draw an edge $T_1 \rightarrow T_2$.

Rule 2 : If T_1 has **Write** operation and then T_2 has **Read** operation then draw an edge $T_1 \rightarrow T_2$.

Rule 3 : If T_1 has **Write** operation and then T_2 has **Write** operation then draw an edge $T_1 \rightarrow T_2$.

Let us draw wait-for graph

Step 1 : Draw vertices for all the transactions



Step 2 : We find the Read-Write pair from two different transactions reading from top to bottom. If such as pair is found then we will add the edges between corresponding directions. For instance -

T_1	T_2
R(A)	
	R(A)
W(A)	
R(B)	
	W(A)
W(B)	

Step 3 :

T_1	T_2
R(A)	
	R(A)
W(A)	
R(B)	
	W(A)
W(B)	

As **cycle is detected** in the wait-for graph there is no need to further process. The **deadlock** is present in this transaction scenario.

Example 7.9.1 *Give an example of a scenario where two phase locking leads to deadlock.*

Solution : Following scenario of execution of transactions can result in deadlock.

T_1	T_2
Lock – S (A)	
	Lock – S (B)
	Read (B)
Read (A)	
Lock – X (B)	
	Lock – X (A)

These two instructions cause a deadlock situation.

In above scenario, transaction T_1 makes an exclusive lock on data item B and then transaction T_2 makes an exclusive lock on data item A. Here unless and until T_1 does not give up the lock (i.e. unlock) on B; T_2 cannot read / write it. Similarly unless and until T_2 does not give up the lock on A; T_1 cannot read or write on A.

This is a purely deadlock situation in two phase locking.

Review Question

1. *What is deadlock ? When it occurs and how to avoid it ?* **GTU : Dec.-10, Marks 7**

7.10 Multi-version and Optimistic Concurrency Control Schemes

- Normally the serializability can be maintained by either delaying an operation or aborting the transaction that issued the operation. For example, a read operation may be delayed because the appropriate value has not been written yet; or it may be rejected. Sometimes the value that was supposed to read has already been overwritten.
- These difficulties can be avoided if old copies of each data item were kept in system.
- In multi-version concurrency control scheme, each Write(X) operation creates a new version of data item X. When transaction issues Read(X) operation, the concurrency control manager selects one of the version of X to be read.
- The concurrency control scheme must ensure that the version to be read is selected in such a manner that the serializability is ensured.

7.10.1 Multi-version Timestamp Ordering

- Assume X_1 , X_2 , ..., X_n are the version of a data item X created by a write operation of transactions.
- Note that the new version of X_i is created only by a write operation.
- With each X_i a RTS (read timestamp) and a WTS (write timestamp) are associated.
- Let,
  - **RTS(X_i) :** The read timestamp of X_i is the largest of all the timestamps of transactions that have successfully read version X_i
  - **WTS(X_i) :** The write timestamp of X_i is the largest of all the timestamps of transactions that have successfully written the value of version X_i.
- To ensure serializability following rules are used -

Rule 1 : If transaction T issues read(X), find the version i of X that has the **highest WTS(X_i)** of all versions of X that is also **less than or equal to TS(T),** then accept the read and update the RTS(X) respectively.

Rule 2 : If transaction T issues write(X) then find the version i of X has the highest WTS(X_i) of all versions of X that is also less than or equal to TS(T), and TS(T) < RTS(X_i), then abort and roll-back T;

Advantage :

(1) Serializability is maintained

Disadvantages :

(1) More storage is needed.

(2) To check unlimited growth of versions, a garbage collection is run periodically.

7.10.2 Optimistic Concurrency Control Scheme

- This is a concurrency technique which assumes that the **conflicts are less** and therefore it is called optimistic concurrency control scheme.
- The basic idea of optimistic concurrency control scheme is that - a schedule is checked against serializability only at the time of commit operation. If non serializable schedules occur then the transactions are aborted.
- This control scheme works in three phases during transaction processing –

(1) **Read phase :** during transaction processing all necessary values are read and stored in local variables. All the updates are made on these local variables only and not on database.

(2) **Validation phase :** After reading phase comes a validation phase. During this phase, it is checked if the values stored in in the local variables can maintain the serializability if we update them to actual database. If any interference occur with these values of local variables. If there is serializability is ensured with these values present in the local variables then the transaction proceed for write phase.

(3) **Write phase :** This phase occurs only after successful validation phase. In this phase the updated values present in the local variables will be written to the database.

Advantages

(1) It is efficient

(2) Serializability is maintained.

Disadvantages

(1) It has to process all three phases, for each transaction.

(2) More storage is needed to maintain data in local variables.

7.11 Database Recovery

GTU : Dec.-06, June-08, March-10, May-12, Summer-13,14,18, Winter-14, 18, Marks 7

- An integral part of a database system is a recovery scheme that can **restore the database** to the consistent state that existed before the failure.
- The recovery scheme must also provide **high availability**; that means, it must minimize the time for which the database is not usable after a failure.

7.11.1 Failure Classification

Various types of failures are -

1) **Transaction failure :** Following are two types of errors due to which the transaction gets failed.
- **Logical error :**
  i) This error is caused due to internal conditions such as bad input, data not found, overflow of resource limit and so on.
  ii) Due to logical error the transaction can not be continued.
- **System error :**
  i) When the system enters in an undesired state and then the transaction can not be continued then this type of error is called as system error.

2) **System crash :** The situation in which there is a hardware malfunction, or a bug in the database software or the operating system, and because of which there is a loss of the content of volatile storage, and finally the transaction processing come to a halt is called system crash.

3) **Disk failure :** A disk block loses its content as a result of either a head crash or failure during a data-transfer operation. The backup of data is maintained on the secondary disks or DVD to recover from such failure.

7.11.2 Storage

A DBMS stores the data on external storage because the amount of data is very huge and must persist across program executions.

The storage structure is a memory structure in the system. It has following categories -

1) Volatile :
- Volatile memory is a primary memory in the system and is placed along with the CPU.
- These memories can store only small amount of data, but they are very fast. For example - main memory, cache memory.
- A volatile storage cannot survive system crashes.
- That means data in these memories will be lost on failure.

2) Non Volatile :
- Non volatile memory is a secondary memory and is huge in size. For example : Hard disk, Flash memory, magnetic tapes.
- These memories are designed to withstand system crashes.

3) Stable :
- Information residing in stable storage is never lost.
- To implement stable storage, we replicate the information in several nonvolatile storage media (usually disk) with independent failure modes.

Stable Storage Implementation
- Stable storage is a kind of storage on which the information residing on it is never lost.
- Although stable storage is theoretically impossible to obtain it can be approximately **built** by applying a technique in which **data loss is almost impossible.**
- That means the information is **replicated** in several nonvolatile storage media with **independent failure modes.**
- Updates must be done with care to ensure that a failure during an update to stable storage **does** not cause a loss of information.

7.11.3 Recovery with Concurrent Transactions

There are four ways for recovery with concurrent transactions :

1) **Interaction with concurrency control :** In this scheme recovery depends upon the concurrency control scheme which is used for the transaction. If a transaction gets failed, then rollback and undo all the updates performed by transaction.

2) **Transaction Rollback :** In this scheme, if the transaction gets failed, then the failed transaction can be rolled back with the help of **log.** The system scans the log backward, and with the help of log entries the system can restore the data items.

3) **Checkpoints :** The checkpoints are used to reduce the number of log records.

4) **Restart recovery :** When the system recovers from the crash it constructs two lists : Undo-list and Redo-list. The undo list consists of transactions to be undone. The redo list consists of transactions to be redone. These two lists are handled as follows

Step 1 : Initially both the lists are empty.

Step 2 : The system scans the log entries from backwards. It scans each entry until it finds **first checkpoint record.**

7.11.4 Shadow Copy Technique

- In the **shadow-copy scheme**, a transaction that wants to update the database first creates a complete **copy of the database.**
- All updates are done on the new database copy, leaving the original copy, the shadow copy, untouched.
- If at any point the transaction has to be aborted, the system merely deletes the new copy. The old copy of the database has not been affected.
- The current copy of the database is identified by a pointer, called **db-pointer**, which is stored on disk.
- If the transaction **partially commits**, it is committed as follows :
  - First, the operating system is asked to make sure that all pages of the new copy of the database have been written out to disk.
  - After the operating system has written all the pages to disk, the database system updates the pointer **db-pointer** to point to the new copy of the database
  - The new copy then becomes the current copy of the database.
  - The old copy of the database is then deleted.
  - The transaction is said to have been **committed** at the point where the **updated db-pointer** is written to disk.
  - The disk system guarantees that it will update **db-pointer** atomically, as long as we make sure that **db-pointer** lies entirely in a single sector.

Where is shadow copy technique used ?

- Shadow copy schemes are commonly used by text editors.
- Shadow copying can be used for small databases.

7.11.5 Log Based Recovery Approach

- Log is the most commonly used structure for recording the modifications that as to be made in the actual database. Hence during the recovery procedure a **log file** is maintained.
- A log record maintains four types of operations. Depending upon the type of operations there are four types of log records-
  1. **<Start>** Log record: It is represented as <T_i, Start>
  2. **<Update>** Log record
  3. **<Commit>** Log record : It is represented as <T_i, Commit>
  4. **<Abort>** Log record : It is represented as <T_i, Abort>

- The log contains various fields as shown in following Fig. 7.11.1. This structure is for <update> operation

Transaction ID(T_i)	Data Item Name	Old Value of Data Item	New Value of Data Item

- For example : The sample log file is

<T_1, Start>
<T_1,a,10,20>
<T_1, Commit>

Here 10 represents the old value before commit operation and 20 is the new value that needs to be updated in the database after commit operation

Fig. 7.11.1 Sample log file

- The log must be maintained on the stable storage and the entries in the log file are maintained before actually updating the physical database.
- There are two approaches used for log based recovery technique - Deferred Database Modification and Immediate Database Modification.

1. Deferred Database Modification :

- In this technique, the database is not updated immediately.
- Only log file is updated on each transaction.
- When the transaction reaches to its commit point, then only the database is physically updated from the log file.
- In this technique, if a transaction fails before reaching to its commit point, it will not have changed database anyway. Hence there is no need for the UNDO operation. The REDO operation is required to record the operations from log file to physical database. Hence deferred database modification technique is also called as **NO UNDO/REDO algorithm.**
- **For example :**

Consider two transactions T_1 and T_2 as follows :

T_1	T_2
Read (A, a)	Read (C, c)
a = a – 10	c = c – 20
Write (A, a)	Write (C, c)
Read (B, b)	
b = b + 10	
Write (B, b)	

If T_1 and T_2 are executed serially with initial values of A = 100, B = 200 and C = 300, then the state of log and database if crash occurs

a) Just after write (B, b)
b) Just after write (C, c)
c) Just after <T_2, commit>

The result of above 3 scenarios is as follows :

Initially the log and database will be

Log	Database
<T_1, Start>	
<T_1, A, 90>	
<T_1, B, 210>	
<T_1, Commit>	
	A = 90
	B = 210
<T_2, Start>	
<T_2, C, 280>	
<T_2, Commit>	
	C = 280

a) Just after write (B, b)

Just after write operation, no commit record appears in log. Hence no write operation is performed on database. So database retains only old values. Hence A = 100 and B = 200 respectively.

Thus the system comes back to original position and no redo operation take place.

The incomplete transaction of T_1 can be deleted from log.

b) Just after write (C, c)

The state of log records is as follows

Note that crash occurs before T_2 commits. At this point T_1 is completed successfully, so new values of A and B are written from log to database. But as T_2 is not committed, there is no redo (T_2) and the incomplete transaction T_2 can be deleted from log.

The redo (T_1) is done as < T_1, commit> gets executed. Therefore A = 90, B = 210 and C = 300 are the values for database.

c) Just after < T_2, commit>

The log records are as follows :

< T_1, Start>
< T_1, A, 90>
< T_1, B, 210>
< T_1, Commit>
< T_2, Start>
< T_2, 6, 280>
< T_2, Commit>

⟵ **Crash occurs here**

Clearly both T_1 and T_2 reached at commit point and then crash occurs. So both redo (T_1) and redo (T_2) are done and updated values will be A = 90, B = 210, C = 280.

2. Immediate Database Modification :

In this technique, the database is updated during the execution of transaction even before it reaches to its commit point.

If the transaction gets failed before it reaches to its commit point, then the a **ROLLBACK Operation** needs to be done to bring the database to its earlier consistent state. That means the effect of operations need to be undone on the database. For that purpose both Redo and Undo operations are both required during the recovery. This technique is known as **UNDO/ REDO** technique.

For example : Consider two transaction T_1 and T_2 as follows :

T_1	T_2
Read(A,a)	Read(C, c)
a = a –10	c = c – 20
Write(A, a)	Write(C, c)
Read(B, b)	
b = b+10	
Write(B, b)	

Here T_1 and T_2 are executed serially. Initially A = 100, B = 200 and C = 300

If the crash occurs after

i) Just after Write(B, b) **ii) Just after Write(C, c)** **iii) Just after <T_2,Commit>**

Then using the immediate Database modification approach the result of above three scenarios can be elaborated as follows :

The contents of **log** and **database** is as follows :

Log	Database
<T_1,Start>	
<T_1,A,100,90>	A = 90
<T_1,B,200,210>	B = 210
<T_1,Commit>	
<T_2,Start>	
<T_2,C,300,280>	
<T_2,Commit>	C = 280

The recovery scheme uses two recovery techniques -

i) **UNDO (T_i) :** The transaction T_i needs to be undone if the log contains <T_i,Start> but does not contain <T_i,Commit>. In this phase, it restores the values of all data items updated by T_i to the old values.

ii) **REDO (T_i) :** The transaction T_i needs to be redone if the log contains both <T_i,Start> and <T_i,Commit>. In this phase, the data item values are set to the new values as per the transaction. After a failure has occurred log record is consulted to determine which transaction need to be redone.

a) Just after Write (B, b) : When system comes back from this crash, it sees that there is <T_1, Start> but no <T_1, Commit>. Hence T_1 must be undone. That means old values of A and B are restored. Thus old values of A and B are taken from log and both the transaction T_1 and T_2 are re-executed.

b) Just after Write (C, c): Here both the redo and undo operations will occur.

c) Undo : When system comes back from this crash, it sees that there is <T_2, Start> but no <T_2, Commit>. Hence T_2 must be undone. That means old values of C is restored. Thus old value of C is taken from log and the transaction T_2 is re-executed.

d) Redo : The transaction T_1 must be done as log contains both the <T_1,Start> and <T_1,Commit> .

So A = 90, B = 210 and C = 300

e) Just after <T_2, Commit> : When the system comes back from this crash, it sees that there are two transaction T_1 and T_2 with both start and commit points. That means T_1 and T_2 need to be redone. So A = 90, B = 210 and C = 280

Example 7.11.1 *Suppose there is a database system that never fails. Is a recovery manager require for this system? Why ?* **GTU : Dec.-05, 06, Marks 3**

Solution :

(1) Yes. Even-though the database system never fails, the recovery manager is required for this system.

(2) During the transaction processing some transactions might be aborted. Such transactions must be rolled back and then the schedule is continued further.

(3) Thus to perform the rollbacks of aborted transactions recovery manager is required.

Review Questions

1. *What is redo and undo operation ?* **GTU : Dec.-06, Marks 5**
2. *Explain system recovery procedure with checkpoint record.* **GTU : Summer-13, Marks 7**
3. *Explain log based recovery and mention all its types.* **GTU : May-12, Marks 3**
4. *Explain immediate database modification log based recovery method. Also explain role of check point in log base.* **GTU : March-10, Marks 7, Winter-14, Marks 5**
5. *What is log based recovery ? Explain immediate database modification technique for database recovery .* **GTU : Dec.-06, Marks 5, June-08, Marks 4, Summer-18, Winter-18, Marks 3**
6. *Explain log based recovery and shadow paging technique.* **GTU : Summer-14, Marks 7**

7.12 Oral Questions and Answers

Q.1 What is a transaction ?

Ans. : A transaction can be defined as a group of tasks that form a single logical unit.

Q.2 What does time to commit mean ?

Ans. : • The COMMIT command is used to save permanently any transaction to database.

• When we perform, Read or Write operations to the database then those changes can be undone by rollback operations. To make these changes permanent, we should make use of commit

Q.3 What are the various properties of transaction that the database system maintains to ensure integrity of data.

OR What are ACID properties ?

Ans. : In a database, each transaction should maintain ACID property to meet the consistency and integrity of the database. These are

(1) Atomicity (2) Consistency (3) Isolation (4) Durability

Q.4 Give the meaning of the expression ACID transaction.

Ans. : The expression ACID transaction represents the transaction that follows the ACID Properties.

Q.5 State the atomicity property of a transaction.

Ans. : This property states that each transaction must be considered as a single unit and must be completed fully or not completed at all.

No transaction in the database is left half completed.

Q.6 What is meant by concurrency control ?

Ans. : A mechanism which ensures that simultaneous execution of more than one transactions does not lead to any database inconsistencies is called concurrency control mechanism.

Q.7 What is the need for concurrency control ?

Ans. : Following are the purposes of concurrency control –

- To ensure isolation
- To resolve read-write or write-write conflicts
- To preserve consistency of database

Q.8 List commonly used concurrency control techniques.

Ans. : The commonly used concurrency control techniques are –

i) Lock ii) Timestamp iii) Snapshot Isolation

Q.9 What is meant by serializability ? How it is tested ?

Ans. : Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.

It is tested using precedence graph technique.

Q.10 What is serializable schedule ?

Ans. : The schedule in which the transactions execute one after the other is called serial schedule. It is consistent in nature. For example : Consider following two transactions T_1 and T_2

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

All the operations of transaction T_1 on data items A and then B executes and then in transaction T_2 all the operations on data items A and B execute. The R stands for read operation and W stands for write operation.

Q.11 When are two schedules conflict equivalent ?

Ans. : Two schedules are conflict equivalent if :

- They contain the same set of the transaction.
- Every pair of conflicting actions is ordered the same way.

Q.12 Define two phase locking.

Ans. : The two phase locking is a protocol in which there are two phases :

i) **Growing Phase (Locking Phase) :** It is a phase in which the transaction may obtain locks but does not release any lock.

ii) **Shrinking Phase (Unlocking Phase) :** It is a phase in which the transaction may release the locks but does not obtain any new lock.

Q.13 What is the difference between shared lock and exclusive lock ?

Ans. :

Shared Lock	Exclusive Lock
Shared lock is used for when the transaction wants to perform read operation.	Exclusive lock is used when the transaction wants to perform both read and write operation.
Multiple shared lock can be set on a transactions simultaneously.	Only one exclusive lock can be placed on a data item at a time.
Using shared lock data item can be viewed.	Using exclusive lock data can be inserted or deleted.

Q.14 What type of lock is needed for insert and delete operations.

Ans. : The exclusive lock is needed to insert and delete operations.

Q.15 What benefit does strict two-phase locking provide ? What disadvantages result ?

Ans. : Benefits :

1. This ensure that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits and preventing other transaction from reading that data .
2. This protocol solves dirty read problem.

Disadvantage :

1. Concurrency is reduced.

Q.16 What is rigorous two phase locking protocol ?

Ans. : This is stricter two phase locking protocol. Here all locks are to be held until the transaction commits.

Q.17 Differentiate strict two phase locking and rigourous two phase locking protocol.

Ans. : • In Strict two phase locking protocol all the exclusive mode locks be held until the transaction commits.

• The rigourous two phase locking protocol is stricter than strict two phase locking protocol. Here all locks are to be held until the transaction commits.

Q.18 Define deadlock.

Ans. : Deadlock is a situation in which when two or more transactions have got a lock and waiting for another locks currently held by one of the other transactions.

Q.19 List four conditions for deadlock.

Ans. :

1. Mutual exclusion condition
2. Hold and wait condition
3. No preemption condition
4. Circular wait condition

Q.20 Why is recovery needed ?

Ans. : • A recovery scheme that can **restore the database** to the consistent state that existed before the failure.

• Due to recovery mechanism, there is high availability of database to its users.

❑❑❑

8 Database Security

Syllabus

Authentication, Authorization and access control, DAC, MAC and RBAC models, Intrusion detection, SQL injection.

Contents

8.1 Basics of Database Security

SPPU : May-11, Marks 2

Definition : Database security is a technique that deals with protection of data against unauthorized access and protection.

Database security is an important aspect for any database management system as it deals with sensitivity of data and information of enterprise.

Database security allows or disallows users from performing actions on the database objects.

8.1.1 Types of Security

Database security addresses following issues –

(1) **Legal Issues :** There are many legal or ethical issues with respect to **right to access information.** For example – if some sensitive information is present in the database, then it must not be accessed by unauthorized person.

(2) **Policy Issues :** There are some Government or organizational policies that tells us what kind of information should be made available to access publicly.

(3) **System Issues :** Under this issue, it is decided whether security function should be handled at **hardware level** or at **operating system level** or at **database level.**

(4) **Data and User level Issues :** In many organizations, multiple security levels are identified to categorize data and users based on these classifications. The security policy of organization must understand these levels for permitting access to different levels of users.

8.1.2 Threats to Database

Threats to database will result in loss or degradation of data. There are three kinds of loss that occur due to threats to database

(1) Loss of Integrity :

- Database integrity means information must be protected from improper modification.
- Modification to database can be performed by inserting, deleting or modifying the data.
- Integrity is lost if unauthorized changes are made to data intentionally or accidently.
- If data integrity is not corrected and work is continued then it results in inaccuracy, fraud, or erroneous decision.

(2) Loss of Availability :

- Database availability means making the database objects available to authorized users.

(3) Loss of Confidentiality :

- Confidentiality means protection of data from unauthorized disclosure of information.

- The loss of confidentiality results in loss of public confidence, or embarrassment or some legal action against organization.

Example 8.1.1 *What is the difference between security and integrity ?* GTU : May 11, Marks 2

Solution : • Data security is the protection of data against unauthorized access or corruption and is necessary to ensure data integrity.

- Data integrity is a desired result of data security, but the term data integrity refers only to the validity and accuracy of data rather than the act of protecting data.

8.2 Authentication, Authorization and Access Control

SPPU : Winter-12, Summer-17, Marks 7

Authentication

- The client has to establish the identity of the server and the server has to establish the identity of the client. This is done often by means of shared secrets such as password and username combination, Personal Identification Number (PIN), or shared biometric data. This process of establishing identity with DBMS is called authentication
- It can also be achieved by a system of higher authority which has previously established authentication.
- In client-server systems where data is distributed, the authentication may be acceptable from a peer system. Thus authentication may be transmissible from system to system

Authorization

- Authorization is a technique of granting permission to authorized user to carry out particular transaction.
- There are several forms of authorization that can be assigned to the user for accessing database. These are –
  - **Read Access :** This form of authorization allows the user only to read data. User cannot insert, delete or modify data to database.
  - **Update Access :** This form of authorization allows the user only to update data. User delete data from database.
  - **Insert Access :** This form of authorization allows the user only to insert data. User cannot update or delete data from database.
  - **Delete Access :** It allows user to delete data only. User cannot modify the existing data.

- There are different roles in authorization that can be given to users –
  1. **System Administrator -** This is the highest administrative authorization for a user. Users with this authorization can also execute some database administrator commands such as restore or upgrade a database.
  2. **System Control -** This is the highest control authorization for a user. This allows maintenance operations on the database but not direct access to data.
  3. **System Maintenance -** This is the lower level of system control authority. It also allows users to maintain the database but within a database manager instance.
  4. **System Monitor -** Using this authority, the user can monitor the database and take snapshots of it.

Difference between Authentication and Authorization

Authentication	Authorization
It checks identity of the user.	It checks user's privileges or access rights to access resources.
It verifies user's credentials.	It validates user's permission.
It occurs before authorization.	It occurs after authentication.
For example – user can authenticate himself for university examination system by entering login and password.	For example – user can access the set of questions for the quiz based as per the access rights given to him/her.

Access Control

- **Definition :** The most common security problem is unauthorized access to computer system. Generally this access is for obtaining the information or to make malicious changes in the database. The security mechanism of a DBMS must include provisions for restricting access to the database system as a whole. This function, called **access control.**
- The objective of access control system is to control operations executed by user in order to prevent actions that could damage data and resources.
- The usual way of supplying access controls to a database system is dependent on the **granting and revoking of privileges** within the database.
- A privilege allows a user to create or access some database object or to run some specific DBMS utilities.

Concept of audit Trails

- When any user wants to access the database system, the DBA (database administrator) will create an account for him/her. The login id and password will be given to him/her on creating the account.
- The user then enters the corresponding login id and password and start accessing the database system.
- For security purposes a **log is maintained** by a database system which keeps track of all operations on the database that are applied by a certain user throughout each **login session,** which consists of the sequence of database interactions that a user performs from the time of **logging** in to the time of **logging off.**
- It is particularly important to keep track of **update operations** that are applied to the database so that, if the database is tampered with, the DBA can determine which **user did the tampering.**
- If any tampering with the database is suspected, then database audit is performed with the help of this log. Such a log maintained for security purpose is called **audit trail.**
- During this database audit, all the operations are examined. When an illegal or unauthorized operation is found, the DBA can determine the account number used to perform the operation.
- Generally, database audits are important for sensitive database systems.

Uses of Audit trail

(1) For maintaining the security against malicious access to database.

(2) For recovering lost transaction.

Review Questions

1. *What is the difference between authorization and authentication ? Explain the use of audit trail* **GTU : Summer-17, Marks 3**
2. *What is authorization and authentication ? Explain the access control in database* **GTU : Winter-12, Marks 7**

8.3 DAC, MAC and RBAC Models

SPPU : Dec.-09, May-11, Summer-13, Winter-14, Marks 7

- There are three types of access control models -
  - Discretionary Access Control (DAC)
  - Mandatory Access Control (MAC)
  - Role Based Access Control (RBAC)
- Before understanding these access control models let us discuss the **access control mechanism.**

- The DBMS must provides selective access to each relation in the database by granting the privileges. For granting the privileges, the access control mechanism follows an authorization model for discretionary privileges known as the **access matrix model.**
- The access matrix is a table with rows and columns. It defines the access permissions.
  - The rows of a matrix M represent subjects (users, accounts, programs)
  - The columns represent objects (relations, records, columns, views, operations).
  - Each position M(i, j) in the matrix represents the types of privileges (read, write, update) that subject i holds on object j.
  - For example –

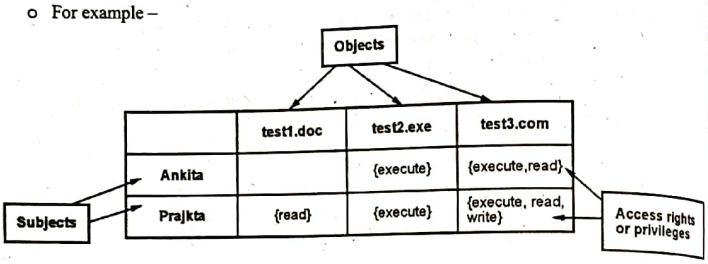


Fig. 8.3.1 Access Matrix Model

8.3.1 Discretionary Access Control (DAC)

- Discretionary Access Control (DAC) allows each **user or subject** to control access to their own data.
- In DAC, owner of resource restricts access to the resources based on the identity of users.
- DAC is typically the default access control mechanism for most desktop operating systems.
- Each resource object on DAC based system has **Account Control List (ACL)** associated with it.
- An ACL contains a list of users and groups to which the user has permitted access together with the level of access for each user or group.
- For example – The ACL is an object centered description of access rights as follows –

> test1.doc: {Prajka: read}
>
> test2.exe: {Ankita: execute}, {Prajkta: execute}
>
> test3.com: {Ankita: execute, read}, {Prajkta: execute, read,write}

- Object access is determined during access control list (ACL) authorization and based on user identification and/or group membership.
- Under DAC a user can only set access permissions for resources which they already own.
- Similarly a hypothetical User A cannot change the access control for a file that is owned by User B. User A can, however, set access permissions on a file that he/she owns.
- User may transfer object ownership to another user(s).
- User may determine the access type of other users.
- The DAC is easy to implement access control model.

Advantages :

(1) It is flexible.

(2) It has simple and efficient access right management.

(3) It is scalable. That means we can add more users without any complexity.

Disadvantages :

(1) It increases the risk that data will be made accessible to users that should not necessarily be given access.

(2) There is no control over information flow as one user can transfer ownership to another user.

8.3.2 Mandatory Access Control (MAC)

- In the mandatory access control, the access control decisions are based on specific relationships between the subject requesting access and the object to which access is requested.
- MAC access control security strategy is generally used in Government and military services.
- MAC takes a **hierarchical approach** to controlling access to resources.
- Under a MAC access to all resource objects is **controlled by** settings defined by the **system administrator.** Thus in MAC all access to resource objects is strictly controlled by the operating system based on system administrator configured settings.
- It is not possible under MAC environment for users to change the access control of a resource.
- Mandatory Access Control begins with **security labels** assigned to all resource objects on the system.
- These security labels contain following information -
  - A **classification** such as confidential, secret and top secret. The classes are top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest.
  - **Attribute** associated with classification

- In some models, **Tuple Classification (TC)** is added. It provides the classification for each tuple as a whole.
- For example – Consider the relation **Employee**

EmpName	Salary	Performance	TC
Ram U	5000 C	Fair S	S
Shyam C	10000 S	Good C	S

- When a person or device tries to access a specific resource, the OS or security kernel will check the entity's credentials to determine whether access will be granted.

Advantages

1. MAC provides tighter security because only a system administrator may access or alter controls.
2. MAC policies reduce security errors.

Disadvantage

1. MAC requires careful planning and continuous monitoring to keep all resource objects' and users' classifications up to date.

Difference between DAC and MAC

Sr.No.	Discretionary Access Control (DAC)	Mandatory Access Control (MAC)
1.	In DAC, owner of resource restricts access to the resources based on the identity of users.	In a MAC access control access to all resource objects is controlled by settings defined by the system administrator. Thus in MAC all access to resource objects is strictly controlled by the operating system based on system administrator configured settings.
2.	In this access control mechanism the key role is played by **resource owner.**	In this access control mechanism the key role is played by **system administrator.**
3.	In this **mechanism Access Control List(ACL)** is used.	In this mechanism **security labels** are used.
4.	DAC is typically the default access control mechanism for most desktop operating systems.	MAC access control security strategy is generally used in Government and military services.
5.	It is more **flexible.**	It is **less flexible.**
6.	It is **less secure.**	It is **more secure.**
7.	It is **easy to implement.**	It is **complex to implement.**

8.3.3 Role Based Access Control (RBAC)

- It is based on the concept that privileges and other permissions are associated with organizational roles, rather than individual users. Individual users are then assigned to appropriate roles.
- For example, an accountant in a company will be assigned to the **Accountant role,** gaining access to all the resources permitted for all accountants on the system. Similarly, a software engineer might be assigned to the **Developer role.**
- In an RBAC system, the roles are centrally managed by the administrator. The administrators determine what roles exist within their companies and then map these roles to job functions and tasks.
- Roles can effectively be implemented using security groups. The security groups are created representing each role. Then permissions and rights are assigned to these groups. Next, simply add the appropriate users to the appropriate security groups, depending on their roles or job functions.
- A user can have more than one role. And more than one user can have the same role.
- Role hierarchies can be used to match natural relations between roles. For example - A Lecturer can create a role Student and give it a privilege "read course material".
- Role Based Access Control (RBAC), also known as **Non discretionary Access Control.**
- RBAC security strategy is widely used by most organizations for deployment of commercial and off-the-shelf products

Advantages :

(1) The security is more easily maintained by limiting unnecessary access to sensitive information based on each user's established role within the organization.

(2) All the roles can be aligned with the organizational structure of the business and users can do their jobs more efficiently and autonomously.

Disadvantages :

(1) It is necessary to understand each user's functionality in depth so that roles can be properly assigned.

(2) If roles are not assigned properly then inappropriate access right creates security severe problems for database system.

Review Questions

1. *Explain the difference between Discretionary Access Control and Mandatory Access Control* **GTU : May-11, Winter-14, Marks 3**
2. *Explain in detail Discretionary Access Control and Mandatory Access Control.* **GTU : Summer-13, Marks 7**
3. *Explain mandatory access control of database security* **GTU : Dec.-09, Marks 3**

8.4 Intrusion Detection

- Traditional database security mechanisms offer basic security features such as authentication, authorization, access control, data encryption, and auditing. However, these mechanisms do not assure protection against the exploitation of vulnerabilities in database management systems (DBMS) and are very limited in defending data attacks from the inside of the organization.
- There are many cases where execution of malicious sequence of SQL commands can not be detected or avoided.
- **Definition :** Intrusion detection system detects the malicious activity in the database and notifies the administrator of the system accordingly. Intrusion Detection System is a system in which malicious activity performed by any user or program is logged and can be viewed later by the admin.
- An intrusion detection system (IDS) can be a hardware device or software application that uses intrusion signature detect abnormal activities present in database.
- The intrusion detection system is based on two models – (1) Anomaly model and (2) Misuse model.
- **Anomaly model** analyzes a set of characteristics of the system and compares their behavior with a set of expected values. It reports when the computed statistics do not match the expected measurements. Anomaly detection uses the assumption that unexpected behavior is evidence of an intrusion.
- **Misuse detection** determines whether a sequence of instructions being executed is known to violate the site's security policy. If so, it reports a potential intrusion.
- Modeling of misuse requires knowledge of system vulnerabilities or potential vulnerabilities that attackers attempt to exploit. The intrusion detection system incorporates this knowledge into a rule set. When data is passed to the intrusion detection system, it applies the rule set to the data to determine if any sequences of data match any of the rules. If so, then it reports that a possible intrusion is present.

Features of Intrusion Detection System

1. It monitors and analysis the **user and system activities.**
2. It performs **auditing** of the system files and other configurations and the operating system.
3. It **assesses the integrity of system** and data files
4. It **conducts analysis** of patterns based on known attacks.
5. It **detects errors** in system configuration.
6. It detects and cautions if the system is in danger.

Review Questions

1. *What is intrusion detection system? Enlist its features*
2. *Explain two models of intrusion detection system*

8.5 SQL Injection

- SQL injection is a **type of code injection technique** that might destroy the databases.
- In this technique the malicious code in SQL statement is placed via web page input. These statements control a database server behind a web application.
- Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.
- An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others.

How SQL Injection Works ?

- To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.
- SQL is a query language that was designed to manage data stored in relational databases. You can use it to access, modify, and delete data. Many web applications and websites store all the data in SQL databases. In some cases, you can also use SQL commands to run operating system commands. Therefore, a successful SQL Injection attack can have very serious consequences.

- **Example of SQL Injection**
  - Following is an example of SQL injection vulnerability works around a simple web application having two input fields – one for user name and another for password.
  - This example has a table named users with the columns username and password

```
uname=request.POST['username']
passwd=request.POST['password']
query="SELECT id FROM users WHERE username=' "+ uname +" ' AND
                                  password =' "+ passwd +" '

database.execute(query)
```

  - Here the two input fields – one for user name and another for password is vulnerable to SQL injection.
  - The attacker can attack use these fields and alter the SQL query to get the access to the database.
  - They could use a trick on password field. They could add

    OR 1 = 1

    Statement to the password field.
  - As a result the query would becomes (assuming username as 'user1'and password ='password') -
  - SELECT id FROM users WHERE username='user1' AND password='password' OR 1=1
  - Because of **OR 1 = 1** statement, the WHERE clause returns the first id from the users table no matter what the username and password are. That means even-if we enter any wrong username or password still the query will get executed because of OR 1 = 1 part which comes out to be true.
  - The first id is returned by the above query for **users** table and we know that the first id is normally administrator. In this way, the attacker not only bypasses authentication but also gains administrator privileges.

How to prevent SQL Injection ?

- The only way to prevent SQL injection is to validate every input field.
- Another method is to make use of parameterized query. This parameterized query is called **prepared statement.** By this ways, application code never use the input directly.
- The Web Application Firewalls (WAF) are also used to filter out the SQL.

Review Questions

1. *What is SQL injection ? Explain it with suitable example*
2. *What are the ways by which the SQL injection can be prevented ?*

8.6 Oral Questions and Answers

Q.1 What do you understand by the term - database security ?

Ans. : Database security is a technique that deals with protection of data against unauthorized access and protection.

Database security is an important aspect for any database management system as it deals with sensitivity of data and information of enterprise.

Q.2 Enlist various threats to database

Ans. : 1) Loss of Integrity 2) Loss of Availability 3) Loss of Confidentiality

Q.3 What is authorization ?

Ans. : Authorization is a technique of granting permission to authorized user to carry out particular transaction.

Q.4 What is access control ?

Ans. : The most common security problem is unauthorized access to computer system. Generally this access is for obtaining the information or to make malicious changes in the database. The security mechanism of a DBMS must include provisions for restricting access to the database system as a whole. This function, called access control.

Q.5 Give two uses of audit trail

Ans. : (1) For maintaining the security against malicious access to database.

(2) For recovering lost transaction.

Q.6 Enlist access control models

Ans. : • Discretionary Access Control (DAC)

- Mandatory Access Control (MAC)
- Role Based Access Control (RBAC)

Q.7 What is Discretionary Access Control(DAC) ?

Ans. : • Discretionary Access Control (DAC) allows each user or subject to control access to their own data.

- In DAC, owner of resource restricts access to the resources based on the identity of users.

Q.8 What is Mandatory Access Control (MAC) ?

Ans. : • In the mandatory access control, the access control decisions are based on specific relationships between the subject requesting access and the object to which access is requested.

- MAC access control security strategy is generally used in Government and military services.

Q.9 What is Role Based Access Control (RBAC) ?

Ans. : • It is based on the concept that privileges and other permissions are associated with organizational roles, rather than individual users. Individual users are then assigned to appropriate roles.

- For example, an accountant in a company will be assigned to the Accountant role, gaining access to all the resources permitted for all accountants on the system. Similarly, a software engineer might be assigned to the Developer role.

Q.10 What is intrusion detection system ?

Ans. : Intrusion detection system detects the malicious activity in the database and notifies the administrator of the system accordingly. Intrusion Detection System is a system in which malicious activity performed by any user or program is logged and can be viewed later by the admin.

Q.11 Enlist at least two features of intrusion detection system

Ans. : 1. It monitors and analysis the user and system activities.

2. It performs auditing of the system files and other configurations and the operating system.

3. It assesses the integrity of system and data files

Q.12 What is SQL Injection ?

Ans. : • SQL injection is a type of code injection technique that might destroy the databases.

- In this technique the malicious code in SQL statement is placed via web page input. These statements control a database server behind a web application.
- Attackers can use SQL Injection vulnerabilities to bypass application security measures.

❑❑❑

9 SQL Concepts

Syllabus

Basics of SQL, DDL,DML,DCL, structure – creation, alteration, defining constraints – Primary key, foreign key, unique, not null, check, IN operator, aggregate functions, Built-in functions –numeric, date, string functions, set operations, sub-queries, correlated sub-queries, join, Exist, Any, All , view and its types., transaction control commands.

Contents

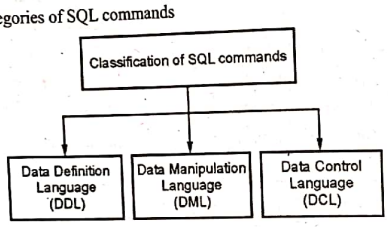
9.1 Basics of SQL

- SQL stands for Structured Query Language.
- It is the language of databases and almost all companies use databases to store their data.
- SQL makes use of query. A Query is a set of instruction given to the database management system. It tells any database what information we would like to get from the database.
- SQL is **case- insensitive**. However it is become standard in SQL community to use **all capital** letters for SQL keywords.
- SQL is a standard language for Relational Database Management System (RDBMS).
- There are various RDBMS software that are popularly used. It includes MySQL, Oracle, MS ACCESS, Microsoft SQL Server, Sybase and so on.

9.2 DDL, DML, DCL Structure

GTU : June-05, Dec.-06, 09, 10, March-10, May-11, 12, Winter-12, 13, 14, 15, 17, 18, Summer-13, 17, 18, Marks 10

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.

9.2.1 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

9.2.2 Insertion

We can insert data into the table using INSERT statement.

Syntax

INSERT INTO table_name (col_1,col_2,...,col_n)
VALUES ($value_1$,$value_2$,...,$value_n$)

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

9.2.3 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

9.2.4 Where

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

9.2.5 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

Person_details 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 query

```
UPDATE person_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	Chennai
444	JJJ	KKK	LLL	Viman nagar	Mumbai

9.2.6 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;
```

9.2.7 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

9.2.8 Order By

- 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

9.2.9 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.

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

9.2.10 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 **perons_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

9.2.11 Aggregate Functions

- An aggregate function allows you to perform a calculation on a set of values to return a single scalar value.
- SQL offers five built-in aggregate functions :
  1. Average : avg
  2. Minimum : min
  3. Maximum : max
  4. Total: sum
  5. Count :

6. The aggregate functions that accept an expression parameter can be modified by the keywords DISTINCT or ALL. If neither is specified, the result is the same as if ALL were specified.

DISTINCT	Modifies the expression to include only distinct values that are not NULL
ALL	Includes all rows where expression is not NULL

7. Syntax of all the Aggregate Functions

```
AVG( [ DISTINCT | ALL ] expression)
COUNT(*)
COUNT( [ DISTINCT | ALL ] expression )
MAX( [ DISTINCT | ALL ] expression)
MIN( [ DISTINCT | ALL ] expression)
SUM( [ DISTINCT | ALL ] expression)
```

8. The **avg** function is used to compute average value. For example – To compute average marks of the students we can use

SQL Statement

```
SELECT AVG(marks)
FROM Students
```

9. The **Count** function is used to count the total number of values in the specified field. It works on both numeric and non-numeric data type. COUNT (*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table. COUNT (*) also considers Nulls and duplicates. For example Consider following table

Test

id	value
11	100
22	200
33	300
NULL	400

SQL Statement

```
SELECT COUNT(*)
FROM Test
```

Output

4

```
SELECT COUNT(ALL id)
FROM Test
```

Output

3

10. The **min** function is used to get the minimum value from the specified column. For example – Consider the above created **Test** table

SQL Statement
SELECT Min(value)
FROM Test
Output
100

11. The **max** function is used to get the maximum value from the specified column. For example – Consider the above created **Test** table

SQL Statement
SELECT Max(value)
FROM Test
Output
400

12. The **sum** function is used to get total sum value from the specified column. For example – Consider the above created **Test** table

SQL Statement
SELECT sum(value)
FROM Test
Output
1000

9.2.12 Group By and Having Clause

(i) Group By :

- The GROUP BY clause is a SQL command that is used to group rows that have the same values.
- The GROUP BY clause is used in the SELECT statement.
- Optionally it is used in conjunction with aggregate functions.
- The queries that contain the GROUP BY clause are called grouped queries
- This query returns a single row for every grouped item.
- **Syntax :**

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
```

- **Example :** Consider the Student table as follows -

sid	sname	marks	city
1	AAA	60	Pune

2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai

Query : Find the total marks of each student in each city

```
SELECT SUM(marks), city
FROM Student
GROUP BY city
```

The result will be as follows –

SUM(marks)	city
150	Pune
125	Mumbai

(ii) Having :

- HAVING filters records that work on summarized GROUP BY results.
- HAVING applies to summarized group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria will be returned.
- HAVING requires that a GROUP BY clause is present.
- WHERE and HAVING can be in the same query.
- **Syntax :**

```
SELECT column-names
 FROM table-name
 WHERE condition
 GROUP BY column-names
 HAVING condition
```

- **Example :** Consider the Student table as follows -

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai
5	EEE	84	Chennai

Query : Find the total marks of each student in the city named 'Pune' and 'Mumbai' only

```
SELECT SUM(marks), city
FROM Student
GROUP BY city
```

```
HAVING city IN('Pune','Mumbai')
```

- The result will be as follows –

SUM(marks)	city
150	Pune
125	Mumbai

Example 9.2.1 *Find the age of youngest sailor with age>=18 for each rating with at least two such sailors. Consider sailor instance as*

sid	sname	rating	age
1	AAA	5	21
2	BBB	2	32
3	CCC	3	19
4	DDD	3	27
5	EEE	10	14
6	FFF	2	47
7	GGG	5	29
8	HHH	4	33
9	III	10	26

Write the SQL query and possible output. Make use of Group By and Having clause

Solution : The query will be

```
SELECT S.rating, MIN (S.age)
        AS Younger
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

The output will be as follows –

rating	Younger
5	21
2	32
3	19

9.2.13 Built-in Functions

In SQL a built-in function is a piece for programming that takes zero or more inputs and returns a value. An example of a built-in functions is ABS(), which when given a value calculates the absolute (non-negative) value of the number.

Query

```
SELECT ABS(-9) Result;
```

Output

Result
9

Mathematical Functions

Function	Value Returned
ABS (m)	Absolute value of m
MOD (m, n)	Remainder of m divided by n
POWER (m, n)	m raised to the nth power
ROUND (m [, n])	m rounded to the nth decimal place
TRUNC (m [, n])	m truncated to the nth decimal place
SIN (n)	sine (n)
COS (n)	cosine (n)
TAN (n)	tan (n)
SQRT (n)	positive square root of n
EXP (n)	e raised to the power n
LOG (n2, n1)	logarithm of n1, base n2
CEIL (n)	smallest integer greater than or equal to n
FLOOR (n)	greatest integer smaller than or equal to n
SIGN (n)	-1 if n < 0, 0 if n = 0, and 1 if n > 0

String Functions

Function	Value Returned
INITCAP (s)	First letter of each word is changed to uppercase and all other letters are in lower case.
LOWER (s)	All letters are changed to lowercase.
UPPER (s)	All letters are changed to uppercase.
CONCAT (s1, s2)	Concatenation of s1 and s2. Equivalent to s1 || s2
LTRIM (s [, set])	Returns s with characters removed up to the first character not in set; defaults to space
RTRIM (s [, set])	Returns s with final characters removed after the last character not in set; defaults to space

REPLACE (s, search_s [, replace_s])	Returns s with every occurrence of search_s in s replaced by replace_s; default removes search_s
SUBSTR (s, m [, n])	Returns a substring from s, beginning in position m and n characters long; default returns to end of s.
LENGTH (s)	Returns the number of characters in s.

For example

```
SELECT CONCAT('Bill', 'Gates') as "NAME"
```

Date and Time Functions

Function	Value Returned
ADD_MONTHS (d, n)	Date d plus n months
LAST_DAY (d)	Date of the last day of the month containing d
MONTHS_BETWEEN (d, e)	Number of months by which e precedes d
NEW_TIME (d, a, b)	The date and time in time zone b when date d is for time zone a
NEXT_DAY (d, day)	Date of the first day of the week after d
SYSDATE	Current date and time
GREATEST (d1, d2, ..., dn)	Latest of the given dates
LEAST (d1, d2, ..., dn)	Earliest of the given dates

For example

```
SELECT SYSDATE();
```

Will result in

2019-06-27 10:02:39

9.2.14 String Operations

- For string comparisons, we can use the comparison operators =, <, >,<=,>=,<> with the ordering of strings determined alphabetically as usual.
- SQL also permits a variety of functions on character strings such as concatenation suing operator||, extracting substrings, finding length of string, converting strings to upper case(using function **upper(s))** and lowercase(using function **lower(s))**, removing spaces at the end of string(using function(trim(s)) and so on.
- Pattern matching can also be performed on strings using two types of special characters –
  - Percent(%): It matches zero, one or multiple characters
  - Underscore(_): The _ character matches any single character.
- The percentage and underscore can be used in combinations.

- Patterns are case sensitive. That means upper case characters do not match lowercase characters or vice versa.
- **For instance :**
  - 'Data%' matches any string beginning with "Data", For instance it could be with "Database", "DataMining","DataStructure"
  - '___' matches any string of exactly three characters.
  - '___%'matches any string of at least length 3 characters.
- The **LIKE** clause can be used in **WHERE** clause to search for specific patterns.
- **For example** – Consider following **Employee Database**

EmpID	EmpName	Department	Date_of_Join
1	Sunil	Marketing	1-Jan
2	Mohsin	Manager	2-Jan
3	Supriya	Manager	3-Jan
4	Sonia	Accounts	4-Jan
5	Suraj	Sales	5-Jan
6	Archana	Purchase	6-Jan

(1) Find all the employee with EmpName starting with "s"

SQL Statement :

```
SELECT * FROM Employee
WHERE EmpName LIKE 's%'
```

Output

EmpID	EmpName	Department	Date_of_Join
1	Sunil	Marketing	1-Jan
3	Supriya	Manager	3-Jan
4	Sonia	Accounts	4-Jan
5	Suraj	Sales	5-Jan

(2) Find the names of employee whose name begin with S and end with a

SQL Statement :

```
SELECT EmpName FROM Employee
WHERE EmpName LIKE 'S%a'
```

Output

EmpName
Supriya
Sonia

(3) Find the names of employee whose name begin with S and followed by exactly four characters

```
SELECT EmpName FROM Employee
WHERE EmpName LIKE 'S____'
```

Output

EmpName
Sunil
Sonia
Suraj

9.2.15 Examples

Example 9.2.2 *For the following database, identify primary key and foreign key where-ever applicable and solve the given queries in SQL.*

Item (ino, description, unit_price)

Supplier (sno, sname, address)

Supplied (sno, ino, sdate, qty, per_unit_discount)

1. Find supplier name for the suppliers who supply every item.

2. Find distinct item names for the items supplied with total discount > 500.

3. Pair of supplier names supplying same items on same dates.

GTU : June-05, Marks 10

Solution :

- The primary key for Item table is **ino**.
- The primary key for supplier table is **sno**
- The primary key for supplied table is **(sno,ino)**

(1) Find supplier name for the suppliers who supply every item.

```
SELECT S.sname
FROM Supplier S
WHERE NOT EXISTS (
      ( SELECT * FROM Item I
        WHERE NOT EXISTS
       ( SELECT *
         FROM Supplied SP
         WHERE SP.sno = S.sno
         AND SP.ino=I.ino )
      )
```


(2) Find distinct item names for the items supplied with total discount > 500.

```
SELECT DISTINCT description
      FROM Item, Supplied
      WHERE Item.ino=Supplied.ino
      AND Supplied.per_unit_discount >500
```

(3) Pair of supplier names supplying same items on same dates.

```
SELECT sname
FROM Supplier
WHERE Supplied.sno=Supplier.sno
      AND
      Item.ino=Supplied.ino
)
```

Example 9.2.3 *Write SQL statements for the following (any five)*

Consider the following database

pilot (pid, pname)

flight (fid, ftype, capacity)

route (pid, fid, from_city, to_city)

i) List the details of flights having capacity more than 300.

ii) List the flights between 'Surat' and 'Mumbai'.

iii) List the names of the pilots who fly from 'Pune'.

iv) List the route on which, pilot named 'Mr Kapoor' flies

v) List the pilots whose names, starts with letter 'A' %' but does not end with letter 'A'.

vi) List the name of pilots who fly 'boing 737' type of flights.

GTU : June-05, Marks 10

Solution :

(i) List the details of flights having capacity more than 300.

```
SELECT * FROM flight
WHERE capacity>300
```

(ii) List the flights between 'Surat' and 'Mumbai'.

```
SELECT fid
FROM flight, route
WHERE
flight.fid=route.fid AND
route.from_city='Surat' AND route.to_city='Mumbai';
```

(iii) List the names of the pilots who fly from 'Pune'.

```
SELECT pname
FROM pilot,route
WHERE pilot.pid=route.pid
AND route.from_city='Pune';
```

(iv) List the route on which, pilot named 'Mr Kapoor' flies

```
SELECT from_city, to_city
FROM route, pilot
WHERE pilot.pid=route.pid
AND pilot.pname='Mr.Kapoor';
```

(v) List the pilots whose names, starts with letter 'A' %' but does not end with letter 'A'.

```
SELECT pname
FROM pilot
WHERE pname LIKE 'A%' MINUS
SELECT pname FROM pilot
WHERE pname LIKE '%A';
```

(vi) List the name of pilots who fly 'boing 737' type of flights.

```
SELECT pname
FROM pilot, flight, route
WHERE flight.ftype = 'boing 737' AND
pilot.pid = route.pid AND
route.fid = flight.fid;
```

Example 9.2.4 *Consider the relation Database.*

Person (SSN, Name, city)

Car (License_no, year, Model, SSN)

Accident(drive_no, SSN, license_no, accidentyear, damage_Amt)

Query :

1. Find out total no of cars that had accident in 1988.

2. Find the Name of driver who did not have an accident in 'Delhi'.

3. Find the car, who don't have total damage of more than ₹1000.

4. Find the cars sold in 2006 and whose owner are from 'Vadodara'

5. How many different models of car are used by 'Mr.abc'

6. Find the lucky persons who have not met any accident yet.

GTU : Dec.-06, Marks 10

Solution :

1. Find out total no of cars that had accident in 1988.

```
SELECT count(License_no)
FROM Car, Accident
WHERE Acccident.accidentyear=1988
```

2. Find the Name of driver who did not have an accident in 'Delhi'.

```
SELECT Name
FROM Person,Accident
WHERE Person.SSN=Accident.SSN AND Peson.city<> 'Delhi'
```


3. Find the car, who don't have total damage of more than ₹ 1000.

```
SELECT License_no
FROM Car, Accident
WHERE Accident.damage_amt<1000 AND
Car.License_no=Accident.licence_no;
```

4. Find the cars sold in 2006 and whose owner are from 'Vadodara'

```
SELECT license_no
FROM Car, Person
WHERE Car.SSN = Person. SSN AND
Car.year = 2006 and Person.city = 'Vadodara';
```

5. How many different models of car are used by 'Mr.abc'

```
SELECT count(model)
FROM Car, Person
WHERE Car.SSN = Person. SSN AND
Person.Name = 'Mr.abc';
```

6. Find the lucky persons who have not met any accident yet.

```
SELECT Name
FROM Person
WHERE person.SSN NOT IN (SELECT SSN FROM Accident);
```

Example 9.2.5 *We have following relations :*

Supplier (S#, sname, status, city)

Parts (P#, pname, color, weight, city)

SP(S#, P#, quantity)

Answer the following queries in SQL,

i) Find name of supplier for city = 'Delhi'.

ii) Find suppliers whose name start with 'AB'.

iii) Find all suppliers whose status is 10, 20 or 30.

iv) Find total number of city of all suppliers.

v) Find s# of supplier who supplies 'red' part.

vi) Count number of supplier who supplies 'red' part.

vii) Sort the supplier table by sname.

GTU : Dec.- 09, Marks 7

Solution :

i) Find name of supplier for city = 'Delhi'.

```
SELECT sname
FROM Supplier
WHERE city='Delhi'
```

ii) Find suppliers whose name start with 'AB'.

```
SELECT sname
FROM Supplier
WHERE sname='AB%'
```

iii) Find all suppliers whose status is 10, 20 or 30.

```
SELECT sname
FROM Supplier
WHERE status BETWEEN 10 AND 30
```

iv) Find total number of city of of all suppliers.

```
SELECT count(*) FROM (SELECT DISTINCT CITY FROM Supplier);
```

v) Find s# of supplier who supplies 'red' part.

```
SELECT DISTINCT supplier.S#
FROM Supplier,Parts, SP
WHERE Supplier.S# = SP.S#
AND SP.P# = Parts.P# AND Parts.color = 'red';
```

vi) Count number of supplier who supplies 'red' part. SELECT count(*)

```
FROM (SELECT DISTINCT Supplier.S#
FROM Supplier,Parts, SP
WHERE Supplier.S# = SP.S#
AND SP.P# = Parts.P# AND Parts.color = 'red');
```

vii) Sort the supplier table by sname.

```
SELECT *
FROM Supplier
ORDER BY sname;
```

Example 9.2.6 *We have following relations :*
Supplier (S#, sname, status, city)
Parts (P#, pname, color, weight, city)
SP(S#, P#, quantity)
Answer the following queries in SQL,
i) Delete records in supplier table whose status is 40.
ii)Add one field in supplier table.

GTU : Dec.-09, Marks 2

Solution :

i) Delete records in supplier table whose status is 40.

```
DELETE FROM Supplier
WHERE status=40
```

ii) Add one field in supplier table.

```
ALTER TABLE Supplier
ADD(PhoneNo number);
```


Example 9.2.7 *We have following relations :*
Supplier (S#, sname, status, city)
Parts (P#, pname, color, weight, city)
SP(S#, P#, quantity)
Answer the following queries in SQL,
i) Find name of parts whose color is 'red'.
ii) Find parts name whose weight less than 10 kg.
iii) Find all parts whose weight from 10 to 20 kg.
iv) Find average weight of all parts.
v) Find S# of supplier who supply part 'p2'.
vi) Find name of supplier who supply maximum parts.
vii) Sort the parts table by pname.

GTU : Dec.-09, Marks 7

Solution :

i) Find name of parts whose color is 'red'.

```
SELECT pname
FROM Parts
WHERE Parts.color = 'red';
```

ii) Find parts name whose weight less than 10 kg.

```
SELECT pname
FROM Parts
WHERE Parts.weight < 10;
```

iii) Find all parts whose weight from 10 to 20 kg.

```
SELECT pname, weight
FROM Parts
WHERE Parts.weight BETWEEN 10 AND 20;
```

iv) Find average weight of all parts.

```
SELECT AVG (weight)
FROM Parts;
```

v) Find S# of supplier who supply part 'p2'.

```
SELECT S#
FROM SP
WHERE p# = 'p2';
```

vi) Find name of supplier who supply maximum parts.

```
SELECT sname, MAX(quantity)
FROM Supplier, Parts,SP
WHERE Supplier.S# = SP.S#
```

```
AND SP.P# = P.P#
GROUP BY sname;
```

vii) Sort the parts table by pname.

```
SELECT *
FROM Parts
ORDER BY pname;
```

Example 9.2.8 *We have following relations :*
Supplier (S#, sname, status, city)
Parts (P#, pname, color, weight, city)
SP(S#, P#, quantity)
Answer the following queries in SQL,
i) Delete records in parts table whose color is 'blue'.
ii) Explain rollback command.

GTU : Dec.-09, Marks 3

Solution : (i) DELETE FROM Parts
WHERE color='blue'

(ii) Refer section 9.4.

Example 9.2.9 *Consider following schema and write SQL for given statements.*
student (rollno, name, branch)
exam(rollno, subject_code, obtained_marks, paper_code)
papers(paper_code, paper_satter_name, university)
i) Display name of student who got first class in subject '130703'.
ii) Display name of all student with their total mark.
iii) Display list number of student in each university.
iv) Display list of student who has not given any exam.

GTU : March-10, Marks 8

Solution :

i) Display name of student who got first class in subject '130703'.

```
SELECT name
FROM student,exam
WHERE exam.obtained_marks>60 AND exam.subject_code='130703' AND
student.rollno=exam.rollno
```

ii) Display name of all student with their total mark.

```
SELECT name,SUM(obtained_marks)
FROM student,exam
WHERE student.rollno=exam.rollno
```

iii) Display list number of student in each university.

```
SELECT COUNT(Rollno)
FROM student, exam, papers
WHERE student.rollno=exam.rollno AND exam.paper_code=papers.paper_code;
```


iv) Display list of student who has not given any exam.

```
SELECT name
FROM student NOT IN
(SELECT rollno
FROM student, exam
WHERE sudent.rollno=exam.rollno)
```

Example 9.2.10 *Write down the query for the following table where primary keys are underlined.*
Person(ss#, name, address)
Car(license, year, model)
Accident(date, driver, damage-amount)
Owns(ss#, license)
Log(license, date, driver)
1) Find the total number of people whose cars were involved in accidents in 2009.
2) Find the number of accidents in which the cars belonging to "S.Sudarshan"
3) Add a new customer to the database.
4) Add a new accident recorded for the santro belonging to "KORTH"

GTU : Dec.-10, Marks 8

Solution :

1) Find the total number of people whose cars were involved in accidents in 2009.

```
SELECT count(ss#)
FROM person ,owns,accident,log
WHERE person.ss#=owns.ss# AND
owns.licence=log.licence AND
log.driver=accident.driver AND
accident.date >= '01-jan-2009' and accident.date < ='31-dec-09';
```

2) Find the number of accidents in which the cars belonging to "S.Sudarshan"

```
SELECT COUNT(accident.date)
FROM person ,owns,accident,log
WHERE log.date=accident.date AND log.driver=accident.driver AND
own.licence=log.licence AND
person.ss#=own.ss# AND
person.name = 's.sudarshan';
```

3) Add a new customer to the database.

```
INSERT INTO person (ss#, name, address) values (101, 'Madhav', 'Pune');
INSERT INTO owns (ss#, license) values (101, 'L101');
INSERT INTO car (license, year, model) values ('L101', 2017, 'Honda');
INSERT INTO log (license, date, driver) values ('L101', NULL, 'Shankar');
```

4) Add a new accident recorded for the santro belonging to "KORTH"

```
INSERT INTO accident (date, driver, damage-amount) VALUES ('31-Dec-2016','Shankar',10000);
```

```
INSERT INTO log (license, date, driver) VALUES ('L111','31-Dec-2016','Shankar');
INSERT INTO car (license, year, model) VALUES ('L111','2005','SANTRO');
INSERT INTO person (ss#, name, address) VALUES (111, 'Korth', 'Mumbai');
INSERT INTO owns (ss#, license) VALUES (111, 'L111');
```

Example 9.2.11 *Consider the employee data.Give an expression in SQL for the following query :*

Employee(employee_name, street,city)
Works(employee_name,company_name,salary)
Company(company_name,city)
Manages(employee_name,manager_name)
1) Find the name of all employees who work for State Bank.
2) Find the names and cities of residence of all employees who work for State Bank.
3) Find all employee in the database who do not work for State Bank.
4) Find all employee in the database who earn more than every employee of UCO bank.

GTU : Dec.-10, Marks 8

Solution :

1) Find the name of all employees who work for State Bank.

```
SELECT employee_name
FROM Works
WHERE company_name='State Bank';
```

2) Find the names and cities of residence of all employees who work for State Bank.

```
SELECT employee_name,city
FROM Employee, Works
WHERE company_name='State Bank' AND
Works.employee_name=Employee.employee_name;
```

3) Find all employee in the database who do not work for State Bank.

```
SELECT employee_name
FROM Works
WHERE company_name< > 'State Bank';
```

4) Find all employee in the database who earn more than every employee of UCO bank.

```
SELECT employee_name from Works
WHERE salary>(SELECT MAX(salary) FROM Works
WHERE company_name='UCO bank');
```

Example 9.2.12 *Consider following schema and write SQL for given statements.*

Student(Rollno,Name,Age,Sex,City).
Student_marks(Rollno,Sub1,Sub2,Sub3,Total,Average)
Write query to
i) Calculate and store total and average marks from sub1, sub2 and sub3.
ii)Display name of students who got more than 60 marks in subject Sub1.

iii)Display name of students with their total and average marks.
iv)Display name of students who got equal marks in subject sub2.

GTU : May-11, Marks 8

Solution :

i) Calculate and store total and average marks from sub1, sub2 and sub3.

```
UPDATE Student_marks
SET Total=sub1+sub2+sub3,
Avarage= (sub1+sub2+sub3)/3;
```

ii) Display name of students who got more than 60 marks in subject Sub1.

```
SELECT Name
FROM Student, Student_marks
WHERE Student.Rollno=Student_marks.Rollno AND Student_marks.sub1>60
```

iii) Display name of students with their total and average marks.

```
SELECT Name, Total, Average
FROM Student, Student_marks
WHERE Student.Rollno=Student_marks.Rollno
```

iv) Display name of students who got equal marks in subject sub2.

```
SELECT Name
FROM Student S, Student_marks SM1, Student_marks SM2
WHERE S.Rollno=SM1.Rollno AND SM1.sub2=SM2.sub2;
```

Example 9.2.13 *We have following relations.*

EMP (empno, ename, jobtitle, manager no, hiredate, sal, comm, dept no)
DEPT (dept no, dname, loc)
i) The employees who are getting salary greater than 3000 for those persons belongings to the department 20
ii) Employees who are not getting any commission.
iii) Find how many job titles are available in employee table.
iv) Display total salary spent for each job category.
v) Display number of employees working in each department and their department name.
vi) List ename whose manager is NULL.
vii) List all employee names and their salaries, whose salary lies between 1500/- and 3500/- both inclusive.

GTU : May-12, Marks 7

Solution :

i) The employees who are getting salary greater than 3000 for those persons belongings to the department 20

```
SELECT ename
```

```
FROM EMP
WHERE EMP.sal>3000 AND EMP.dept_no=20
```

ii) Employees who are not getting any commission.

```
SELECT ename
FROM EMP
WHERE EMP.comm IS NULL;
```

iii) Find how many job titles are available in employee table.

```
SELECT count(jobtitle)
FROM EMP
```

iv) Display total salary spent for each job category.

```
SELECT ename,SUM(sal)
FROM EMP
GROUP BY jobtitle;
```

v) Display number of employees working in each department and their department name.

```
SELECT COUNT(EMP.eno), DEPT.dname
FROM EMP, DEPT
WHERE EMP.dept_no=DEPT.dept_no
GROUP BY DEPT.dept_no;
```

vi) List ename whose manager is NULL.

```
SELECT ename
FROM EMP
WHERE manager_no IS NULL;
```

vii) List all employee names and their salaries, whose salary lies between 1500/- and 3500/- both inclusive.

```
SELECT ename, sal
FROM EMP
WHERE sal>=1500 AND sal<=3500;
```

Example 9.2.14 *We have following relations.*

EMP (empno, ename, jobtitle, manager no, hiredate, sal, comm, dept no)

DEPT (dept no, dname, loc)

Answer the following queries in SQL

i) Find the employees working in the department 10, 20, 30 only.

ii) Find employees whose names start with letter A or letter a.

iii) Find employees along with their department name.

iv) Find employees whose manager is KING.

v) Find the employees who are working in smith's department.

vi) Find the employees who get salary more than Allen's salary.

vii) Display employees who are getting maximum salary in each department.

GTU : May-12, Marks 7

Solution :

i) Find the employees working in the department 10, 20, 30 only.

```
SELECT empno
FROM EMP
WHERE dept_no BETWEEN 10 AND 30
```

ii) Find employees whose names start with letter A or letter a.

```
SELECT ename
FROM EMP
WHERE ename='A%' OR ename='a%';
```

iii) Find employees along with their department name.

```
SELECT EMP.ename,DEPT.dname
FROM EMP, DEPT
WHERE EMP.dept_no=DEPT.dept_no ;
```

iv) Find employees whose manager is KING.

```
SELECT ename
FROM EMP
WHERE managerno = (SELECT empno FROM EMP WHERE ename='KING');
```

v) Find the employees who are working in smith's department.

```
SELECT ename
FROM EMP,DEPT
WHERE EMP.dept_no=DEPT.dept_no AND ename='Smith';
```

vi) Find the employees who get salary more than Allen's salary.

```
SELECT ename
FROM EMP
WHERE sal> (SELECT sal FROM EMP WHERE ename='Allen');
```

vii) Display employees who are getting maximum salary in each department.

```
SELECT ename, MAX(sal)
FROM EMP
GROUP BY dept_no;
```

Example 9.2.15 *Write queries for the following table.*

T1 (Empno, Ename, Salary, Designation),

T2 (Empno, Deptno.)

i) Display all rows for salary greater than 5000

ii) Display the deptno for the ename='syham'.

iii) Add a new column deptname in table T2.

iv) Change the designation of ename='ram' from 'clerk' to 'senior clerk'.

v) Find the total salary of all the rows.

vi) Display Empno, Ename, Deptno and Deptname.
vii) Drop the table T1.

GTU : Winter-12, Marks 7

Solution :

i) Display all rows for salary greater than 5000

```
SELECT *
FROM T1
WHERE Salary>5000
```

ii) Display the deptno for the ename='shyam'.

```
SELECT deptno
FROM T1,T2
WHERE T1.Empno=T2.Empno AND T1.Ename='shyam';
```

iii) Add a new column deptname in table T2.

```
ALTER TABLE T2
ADD (deptname VARCHAR(20));
```

iv) Change the designation of ename='ram' from 'clerk' to 'senior clerk'.

```
UPDATE T1
SET T1.designation='Senior Clerk'
WHERE T1.ename='ram';
```

v) Find the total salary of all the rows.

```
SELECT SUM(Salary)
FROM T1;
```

vi) Display Empno, Ename, Deptno and Deptname.

```
SELECT E.Empno, E.Ename, D.Deptno, D.Deptname
FROM T1 E, T2 D
WHERE E.Empno=D.Empno;
```

vii) Drop the table T1.

```
DROP Table T1;
```

Example 9.2.16 *Solve following queries with following table, where underlined attribute is primery key.*

Person(SS#, name, address)
Car(license, year, model)
Accident(date, driver,damage-amount)
Owns(SS##,license)
Log(licence, date, driver)

i) Find the name of person whose license number is '12345'.
ii) Display name of driver with number of accidents done by that driver.
iii) Add a new accident by 'Ravi' for 'BMW' car on 01/01/2013 for damage amount of 1.5 lakh rupees.

GTU : Winter-13, Marks 7

Solution :

i) Find the name of person whose license number is '12345'.

```
SELECT name
FROM Person, Owns,Car
WHERE Person.SS#=Owns.SS# AND
Car.license=Owns.license AND
Car.license='12345';
```

ii) Display name of driver with number of accidents done by that driver.

```
SELECT driver,COUNT(*)
FROM Accident
GROUP BY driver
```

iii) Add a new accident by 'Ravi' for 'BMW' car on 01/01/2013 for damage amount of 1.5 lakh rupees.

```
INSERT INTO Person (SS#, name, address) VALUES(111,'Ravi','Mumbai');
INSERT INTO Car (license,year,model) VALUES('L111','2008,'BMW');
INSERT INTO Owns(SS#, license) VALUES(111, 'L111');
INSERT INTO Accident('01/01/2013','Ravi',150000);
INSERT INTO Log('L111', '01/01/2013','Ravi');
```

Example 9.2.17 *For Supplier - Parts database*

Supplier(S#, sname, status, city)
Parts(P#, pname,color,weight,city)
SP(S#, P#, quantity)
Answer the following queries in SQL.
i) Display the name of supplier who lives in 'Ahemdabad'.
ii) Display the parts name which is not supplied yet.
iii) Find all suppliers whose status is either 20 or 30.

GTU : Winter 13, Marks 7

Solution :

i) Display the name of supplier who lives in 'Ahemdabad'.

```
SELECT sname
FROM Supplier
WHERE city='Ahmdabad';
```

ii) Display the parts name which is not supplied yet.

```
SELECT pname
FROM Parts, SP
WHRE SP.P# <> Parts.P#
```

iii) Find all suppliers whose status is either 20 or 30.

```
SELECT sname
FROM Supplier
WHERE status = 20 or status = 30;
```

Example 9.2.18 *For Supplier - Parts database*

Supplier(S#, sname, status, city)
Parts(P#, pname,color,weight,city)
SP(S#, P#, quantity)
Answer the following queries in SQL.
i) Find the name of parts having 'Red' colour.
ii) Delete parts whose weight is more than 100 gram.
iii) Count how many times each supplier has supplied part 'P2'.
iv) How much times shipment is for more than 100 quantities ?

GTU : Winter-13, Marks 7

Solution :

i) Find the name of parts having 'Red' colour.

```
SELECT pname
FROM Parts
WHERE color='Red';
```

ii) Delete parts whose weight is more than 100 gram.

```
DELETE FROM Parts
WHERE weight>100
```

iii) Count how many times each supplier has supplied part 'P2'.

```
SELECT S#, COUNT(*)
FROM SP
WHERE P#='P2'
```

iv) How much times shipment is for more than 100 quantities?

```
SELECT S#, COUNT(*)
FROM SP
WHERE quantity>100;
```

Example 9.2.19 *Consider following schema and write SQL for given statements.*

student(RollNo, Name, Age, Sex, City)
Student_marks(RollNo, Sub1, Sub2, Sub3, Total, Average)
Write query to
i) Display name and city of students whose total marks are greater than 225.
ii) Display name of students who got more than 60 marks in each subject.
iii) Display name of city from where more than 10 students come from.
iv) Display a unique pair of male and female students.

GTU : Summer-13, Marks 8

Solution :

i) Display name and city of students whose total marks are greater than 225.

```
SELECT name, city
FROM student,student_marks
WHERE student.RollNo = student_marks.RollNo AND student_marks.Total>225
```

ii) Display name of students who got more than 60 marks in each subject.

```
SELECT name
FROM student,student_marks
WHERE student.RollNo = student_marks.RollNo AND
student_marks.Sub1>60 OR student_marks.Sub2>60 OR student_marks.Sub3>60;
```

iii) Display name of city from where more than 10 students come from.

```
SELECT city
FROM student
WHERE count(RollNo)>10
```

iv) Display a unique pair of male and female students.

```
SELECT S1.name
FROM Student S1, Student S2
WHERE S1.Name=S2.Name AND S1.sex='M' S2.sex='F'
```

Example 9.2.20 *C Write queries for the following tables :*

T1 (Empno, Ename, Salary, Designation)
T2 (Empno, Deptno.)
1) Display all the details of the employee whose salary is lesser than 10 K.
2) Display the Deptno in which employee Seeta is working.
3) Add a new column Deptname in table T2.
4) Change the designation of Geeta from 'Manager' to 'Senior Manager'.
5) Find the total salary of all the employees.
6) Display Empno, Ename, Deptno and Deptname
7) Drop the table T1.

GTU : Winter-14, Marks 7

Solution :

1) Display all the details of the employee whose salary is lesser than 10 K.

```
SELECT *
FROM T1
WHERE Salary<10000
```

2) Display the Deptno in which employee Seeta is working.

```
SELECT Deptno
FROM T2, T1
WHERE T1.Empno=T2.Empno AND T1.Ename='Seeta';
```

3) Add a new column Deptname in table T2.

```
ALTER TABLE T2
ADD (Deptname VARCHAR(20));
```

4) Change the designation of Geeta from 'Manager' to 'Senior Manager'.

```
UPDATE T1
SET Designation ='Senior Manager'
WHERE Ename='Geeta';
```

5) Find the total salary of all the employees.

```
SELECT SUM(Salary)
FROM T1
```

6) Display Empno, Ename, Deptno and Deptname

```
SELECT T1.Empno,T1.Ename, T2.Deptno, T2.Deptname
FROM T1, T2
```

7) Drop the table T1.

```
DROP TABLE T1
```

Example 9.2.21 *We have following relations :*

EMP(empno, ename, jobtitle, managerno, hiredate, sal, comm, deptno)

DEPT(deptno, dname, loc)

Answer the following queries in SQL.

i) Find the Employees working in the department 10, 20, 30 only.

ii) Find Employees whose names start with letter A or letter a.

iii) Find Employees along with their department name.

iv) Insert data in EMP table.

v). Find the Employees who are working in Smith's department

vi) Update Department name of Department No = 10

vii) Display employees who are getting maximum salary in each department

GTU : Winter-15, Marks 7

Solution :

i) Find the Employees working in the department 10, 20, 30 only.

```
SELECT ename
FROM EMP
WHERE deptno BETWEEN 10 AND 30
```

ii) Find Employees whose names start with letter A or letter a.

```
SELECT ename
FROM EMP
WHERE ename='A%' OR ename='a%';
```


iii) Find Employees along with their department name.

```
SELECT EMP.ename,DEPT.dname
FROM EMP, DEPT
WHERE EMP.deptno=DEPT.deptno ;
```

iv) Insert data in EMP table.

```
INSERT INTO EMP( empno, ename, jobtitle, managerno, hiredate, sal, comm, deptno)
VALUES ('E111','AAA','Manager',M123,01-01-2010,20000,2000,'D111');
```

v) Find the Employees who are working in Smith's department

```
SELECT ename
FROM EMP,DEPT
WHERE EMP.deptno=DEPT.deptno AND ename='Smith';
```

vi) Update Department name of Department No = 10

```
UPDATE DEPT
SET dname='Accounts'
WHERE deptno=10;
```

vii) Display employees who are getting maximum salary in each department

```
SELECT ename, MAX(sal)
FROM EMP
GROUP BY deptno;
```

Example 9.2.22 *Consider following Hotel database, primary keys are underlined:*

hotel(hotel_no,name,type,price)

room(room-no,hotel-no,type,price)

booking(hotel-no,guest-no,date-from,date-to,room-no)

guest(guest-no,name,address)

Give an expression in SQL for each of the following queries

(1) List the names and addresses of all guests in London, alphabetically ordered by name.

(2) List out hotel name and total number of rooms available

(3) List the details of all the rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.

(4) List all guests currently staying a the Grosvenor Hotel.

(5) List the rooms that are currently unoccupied at the Grosvenor Hotel.

(6) List the number of rooms in each hotel in London.

(7) List out all guests who have booked room for three or more days.

GTU : Summer-17, Marks 7

Solution :

(1) List the names and addresses of all guests in London, alphabetically ordered by name.

```
SELECT name, address
FROM guest
```

```
WHERE address LIKE '%London%'
ORDER BY name;
```

(2) List out hotel name and total number of rooms available

```
SELECT name, COUNT(room-no)
FROM hotel, room
WHERE hotel.hotel_no=room.hotel_no
GROUP by hotel_no;
```

(3) List the details of all the rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.

```
SELECT r.* FROM Room r LEFT JOIN
(SELECT g.guestName, h.hotelNo, b.roomNo
      FROM guest g, booking b, hotel h
WHERE g.guest_no = b.guest_no AND
b.hotel_no = h.hotel_no AND
h.name= 'Grosvenor Hotel' AND
date_from <= CURRENT_DATE AND
date_to >= CURRENT_DATE)
AS XXX ON r.hotel_no = XXX.hotel_no AND r.room_no = XXX.room_no;
```

(4) List all guests currently staying at the Grosvenor Hotel.

```
SELECT * FROM guest
WHERE guest_no =
(SELECT guest_no FROM booking
WHERE
date-from <= CURRENT_DATE AND date-to >= CURRENT_DATE AND
hotel_no =
(SELECT hotel_no FROM hotel
WHERE name = 'Grosvenor Hotel'));
```

(5) List the rooms that are currently unoccupied at the Grosvenor Hotel.

```
SELECT (r.hotel_no, r.room_no, r.type, r.price)
FROM room r, hotel h
WHERE r.hotel_no = h.hotel_no AND
h.name = 'Grosvenor Hotel' AND
NOT EXIST
 (SELECT *
  FROM booking b, hotel h
  WHERE (date_from <= 'CURRENT_DATE'
   AND date_to >= 'CURRENT_DATE')
   AND r.hotel_no=b.hotel_no
   AND r.room_no=b.room_no
   AND r.hotel_no=h.hotel_no
    AND name = 'Grosvenor Hotel');
```


(6) List the number of rooms in each hotel in London.

```
SELECT hotel_no, COUNT(room_no) AS count
FROM room r, hotel h
WHERE r.hotel_no = h.hotel_no AND city = 'London'
GROUP BY hotel_no;
```

(7) List out all guests who have booked room for three or more days.

```
SELECT guest-no, name
FROM guest g ,booking b
WHERE b.date-to Minus b.date-from >=3
```

Example 9.2.23 *Consider following schema and write SQL for given statements*

employee(employee-name,street,city)

works(employee-name, company-name, salary)

company(company-name, city)

manages(employee-name,manager-name)

(1) Find the names of all employees who work for First Bank Corporation.

(2) Give all employees of First Bank Corporation a 10-percent raise.

(3) Find the names and cities of residence of all employees who work for First Bank Corporation.

(4) Find the names and Street addresses, cities of residence of all employees who work for First Bank Corporation and earn more than $10,000

(5) Find all employees in the database who live in the same cities as the companies for which they work.

(6) Find all employees in the database who do not work for First Bank Corporation.

(7) Find the company and number of employees in company that has more than 30 employees.

GTU : Summer -17,Winter-18, Marks 7

Solution :

(1) Find the names of all employees who work for First Bank Corporation.

```
SELECT employee_name
FROM works
WHERE company_name='First Bank Corporation';
```

(2) Give all employees of First Bank Corporation a 10-percent raise.

```
UPDATE works
SET salary=salary*1.1
WHERE company_name='First Bank Corporation';
```

(3) Find the names and cities of residence of all employees who work for First Bank Corporation.

```
SELECT employee_name,city
FROM employee
WHERE employee_name IN
```

```
(SELECT employee_name
FROM works
WHERE company-name='First Bank Corporation');
```

(4) Find the names and Street addresses, cities of residence of all employees who work for First Bank Corporation and earn more than $10,000

```
SELECT employee_name,city
FROM employee
WHERE employee_name IN
(SELECT employee_name
FROM works
WHERE company-name='First Bank Corporation' AND salary>10000);
```

OR

```
SELECT E.employee_name, E.street, E.city
FROM employee as E, works as W
WHERE
E.employee_name=W.employee_name AND
W.company_name='First Bank Corporation' AND W.salary>10000
```

(5) Find all employees in the database who live in the same cities as the companies for which they work.

```
SELECT E.employee_name
FROM employee as E, works as W, company as C
where E.employee_name=W.employee_name AND
E.city=C.city AND
W.company_name=C.company_name;
```

(6) Find all employees in the database who do not work for First Bank Corporation.

```
SELECT employee_name
FROM works
WHERE company_name <>'First Bank Corporation';
```

(7) Find the company and number of employees in company that has more than 30 employees

```
SELECT company-name,COUNT(employee-name)
FROM employee E, company C,works W
WHERE E.employee-name=W.employee-name AND
W.company-name=C.company-name
HAVING COUNT(employee-name)>30;
```

Example 9.2.24 *Consider following relations and write SQL queries for given statements*
Assume suitable constraints
Instructor(ID, Name, Dept_name, Salary)
Teaches(ID,Course_id, Sec_id, Semester(even/odd), Year)
(1) Write SQL query to create Instructor table

(2) Find the average salary of the instructor in computer department.
(3) Find the number of instructors in each department who teach a course in even semester of 2016
(4) Find the names of instructor with salary amounts between 30000 and 50000

GTU : Winter-17, Marks 7

Solution :

(1) Write SQL query to create Instructor table

```
CREATE TABLE Instructor
(ID CHAR,
Name VARCHAR(20),
Dept_name VARCHAR(15),
Salary numeric(8,2)
);
```

(2) Find the average salary of the instructor in computer department.

```
SELECT AVG(Salary)
FROM Instructor
WHERE Dept_name='Computer';
```

(3) Find the number of instructors in each department who teach a course in even semester of 2016

```
SELECT COUNT(DISTINCT ID)
FROM Teaches
WHERE Semester='even' AND Year=2016
```

(4) Find the names of instructor with salary amounts between 30000 and 50000

```
SELECT Name
FROM Instructor
WHERE Salary >=30000 AND Salary <=50000
```

Example 9.2.25 *Consider following schema and write SQL for given statements*
Client_master(clientno,name,address,city,pincode,state,baldue)
Product_master(productno,name,profitpercent,unitmeasure,sellprice,costprice)
Sales_master(Salesmanno,name,address,city,pincode,state,salary,tgtotget,remarks)
(1) Find out the names of all clients
(2) List all the clients who are located in Mumbai
(3) Delete all salesmen from salesman_master whose salaries are equal to ₹3500
(4) Destroy the table client_master along with data
(5) List the names of all clients having 'a' as the second letter in their names.
(6) Count the number of products having cost price is less than or equal to 500.
(7) Calculate the average, minimum and maximum Sell price of product

GTU : Summer-18, Marks 7

Solution :

```
(1) SELECT name FROM Client_master;
(2) SELECT * FROM Client_master
      WHERE city='Mumbai'
(3) DELETE FROM Salesman_master
      WHERE salary=3500;
(4) DROP TABLE Client_master;
(5) SELECT name
FROM Client_master
WHERE name like '_a%';
(6) SELECT count(productno)
FROM product_master
WHERE costprice <= 500
(7) SELECT AVG(sellprice), MIN(sellprice), MAX(sellprice)
FROM product_master;
```

Example 9.2.26 *Assume the following table.*

Degree (degcode, name, subject)

Candidate (seatno, degcode, name, semester, month, year, result)

Marks (seatno, degcode, semester, month, year, papcode, marks)

[degcode – degree code, name – name of the degree (Eg. MSc.), subject – subject of the course (Eg. Physis), papcode – paper code (Eg. A1)]

Solve the following queries using SQL;

Write a SELECT statement to display,

(i) all the degree codes which are there in the candidate table but not present in degree table in the order of degcode.

(ii) the name of all the candidates who have got less than 40 marks in exactly 2 subjects.

(iii) the name, subject and number of candidates for all degrees in which there are less than 5 candidates.

(iv) the names of all the candidate who have got highest total marks in MSc. Maths.

Solution :

(i) SELECT C.degcode

```
FROM Candidate C,
WHERE NOT EXISTS
        (SELECT  D.degcode
        FROM Degree D
        WHERE D.degcode=C.degcode)
ORDER by C.degcode
```

(ii) SELECT C.name

```
FROM Candidate C, Degree D, Marks M
WHERE
C.seatno=M.seatno AND C.degcode=D.degcode AND C.degcode=M.degcode AND M.marks<40
```



```
GROUP BY C.seatno
HAVING count(D.subject)=2;
```

(iii) SELECT D.name,D.subject,count(*)

```
FROM degree D, Candidate C
WHERE D.degcode=C.degcode
HAVING( SELECT count(*) FROM Candidate <5);
```

(iv) SELECT C.name

```
FROM Candidate C, Degree D, Marks M
WHERE
D.degname='MSc' AND D.subject='Maths' AND C.degcode=D.degcode AND C.seatno=M.seatno
AND
M.marks= (SELECT max(M.marks) FROM Marks M)
```

Example 9.2.27 *Consider a student registration database comprising of the below given table schema.*

Student File

Student Number	Student Name	Address	Telephone

Course File

Course Number	Description	Hours	Professor Number

Professor File

Professor Number	Name	Office

Registration File

Student Number	Course Number	Date

Consider a suitable sample of tuples / records for the above mentioned tables and write DML statements (SQL) to answer for the queries listed below.

i) Which courses does a specific professor teach ?

ii) What courses are taught by two specific professors ?

iii) Who teaches a specific course and where is his/her office ?

iv) For a specific student number, in which courses is the student registered and what is his/her name ?

v) Who are the professors for a specific student ?

vi) Who are the students registered in a specific course ?

Solution :

(i)

```
SELECT P.name,C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
HAVING count(DISTINCT P.name)=2
```

(ii)

```
SELECT P.name,C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
```

(iii)

```
SELECT P.name,P.office, C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
```

(iv)

```
SELECT S.StudentNumber,S.StudentNumber,C.Description
FROM Student S, Course C, Registration R
WHERE S.StudentNumber=R.StudentNumber AND C.CourseNumber=R.CourseNumber
```

(v)

```
SELECT S.StudentName, P.Name
FROM Student S, Course C, Professor P, Registration R
WHERE C.ProfessorNumber=P.ProfessorNumber
      AND C.CourseNumber=R.CourseNumber
      AND S.StudentNumber=R.StudentNumber
      GROUP BY P.ProfessorNumber
```

(vi)

```
SELECT S.StudentName, C.Description
FROM Student S, Course C, Registration R
WHERE S.StudentNumber=R.StudentNumber
AND R.CourseNumber=C.CourseNumber
GROUP BY C.CourseNumber
```

Review Questions

1. *Explain following terms with suitable example.*
   (1) Primary key (2) Candidate key (3) Foreign key (4) Check constraint
2. *Explain aggregate functions of SQL with suitable example.* **GTU : Winter-17, Marks 7**

9.3 GRANT and INVOKE Commands

GTU : Summer-18, Winter-18, Marks 3

(1) GRANT Command

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

Syntax

```
GRANT privilege_name
ON object_name
```



```
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];
```

- privilege_name is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- object_name is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
- user_name is the name of the user to whom an access right is being granted.
- user_name is the name of the user to whom an access right is being granted.
- PUBLIC is used to grant access rights to all users.
- ROLES are a set of privileges grouped together.
- WITH GRANT OPTION – allows a user to grant access rights to other users.

Example

```
GRANT SELECT ON person_details TO user1
```

This query will grant the SELECT permission to person_details table to user named user1.

Similarly we can GRANT more than one privileges to user in a table

```
GRANT SELECT, INSERT,DELETE, UPDATE ON person_details TO user1
```

For granting all privileges to user we use **sysdba**. The sysdba is a set of priviliges which has all the permissions in it.

```
GRANT sysdba TO user1
```

(2) REVOKE

The REVOKE command removes user access rights or privileges to the database objects.

Syntax

```
REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}
```

Example

To remove access right for SELECT to the table **person_details** for user1 we write the query

```
REVOKE SELECT ON person_details FROM user1;
```

Review Question

1. *Describe GRANT and REVOKE commands.* **GTU : Summer-18, Winter-18, Marks 3**

9.4 Transaction Control (TCL) Commands

GTU : Summer-18, Winter-18, Marks 4

The COMMIT, ROLLBACK, and SAVEPOINT are collectively considered as Transaction Commands

(1) **COMMIT :** The COMMIT command is used to save permanently any transaction to database.

When we perform, Read or Write operations to the database then those changes can be undone by rollback operations. To make these changes permanent, we should make use of **commit**

(2) **ROLLBACK :** The ROLLBACK command is used to undo transactions that have not already saved to database. For example

Consider the database table as

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 9.4.1 Student table

Following command will delete the record from the database, but if we immediately performs ROLLBACK, then this deletion is undone.

For instance –

```
DELETE FROM Student
WHERE RollNo =2;
ROLLBACK;
```

Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

(3) **SAVEPOINT :** A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction. The SAVEPOINT can be created as


```
SAVEPOINT savepoint_name;
```

Then we can ROLLBACK to SAVEPOIT as

```
ROLLBACK TO savepoint_name;
```

For example – Consider Student table as follows –

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 9.4.2 Student table

Consider Following commands

```
SQL> SAVEPOINT S1
SQL>DELETE FROM Student
Where RollNo=2;
SQL> SAVEPOINT S2
SQL>DELETE FROM Student
Where RollNo=3;
SQL> SAVEPOINT S3
SQL>DELETE FROM Student
Where RollNo=4
SQL> SAVEPOINT S4
SQL>DELETE FROM Student
Where RollNo=5
SQL> ROLLBACK TO S3;
```

Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC

Thus the effect of deleting the record having RollNo 2, and RollNo3 is undone.

Review Question

1. *Compare rollback with commit SQL commands.* GTU : Summer-18, Winter-18, Marks 4

9.5 Oral Questions and Answers

Q.1 What is SQL ?

Ans. : SQL stands for Structured Query Language. It is the language of databases and almost all companies use databases to store their data. SQL makes use of query. A Query is a set of instruction given to the database management system. It tells any database what information we would like to get from the database.

Q.2 What are three categories of SQL ?

Ans. : 1) DDL : Data Definition Language

2) DML: Data Manipulation Language

3) DCL: Data Control Language

Q.3 What is the purpose of SELECT statement ?

Ans. : The select statement is used to fetch the data from the database table.

Q.4 What is resultset ?

Ans. :

- 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.

Q.5 Give the syntax of UPDATE query

Ans. : UPDATE table_name

SET col1=value1, col2=value2,...

WHERE condition;

Q.6 What is the use of order by tag in SQL ?

Ans. :

- 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.

Q.7 How will you specify NOT NULL constraint ?

Ans. : 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)
    );
```


Q.8 What is the use of IN operator ?

Ans. : The IN operator is just similar to OR operator.

It allows to specify multiple values in WHERE clause.

Q.9 What is count function used in SQL ?

Ans. : The count function is used to count the total number of values in the specified field. It works on both numeric and non-numeric data type. COUNT (*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table.

Q.10 What is the use of Group by clause ?

Ans. : The GROUP BY clause is a SQL command that is used to group rows that have the same values.

The GROUP BY clause is used in the SELECT statement.

Q.11 What is the purpose of GRANT command ?

Ans. : SQL GRANT is a command used to provide access or privileges on the database objects to the users.

Q.12 What is the purpose of REVOKE command ?

Ans. : The REVOKE command removes user access rights or privileges to the database objects.

❑❑❑

Notes

10 PL/SQL Concepts

Syllabus

Cursors, Stored Procedures, Stored Function, Database Triggers

Contents

10.1 Basics of PL/SQL

- PL/SQL stands for Procedural Language extensions to the Structured Query Language (SQL).
- PL/SQL is a combination of SQL along with the procedural features of programming languages.
- It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL.
- Oracle uses a PL/SQL engine to processes the PL/SQL statements.
- PL/SQL includes procedural language elements like conditions and loops. It allows declaration of constants and variables, procedures and functions, types and variable of those types and triggers.

10.2 How to Set Environment for Executing PL/SQL Scripts ?

We need to install Oracle for executing the PL/SQL script. The latest edition of Oracle is 11g. We can install it on Linux as well as on Windows (32 bit or 64 bit) platform

Following steps can be followed for installing the Oracle

Step 1: Open the web page **https://www.oracle.com**.

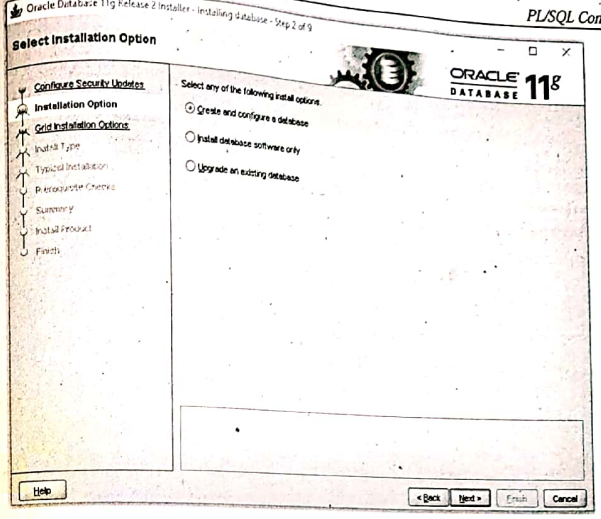
The scroll down to Oracle 11g Release 2 and download two files File1 and File2, for the Operating system on which you work.

Step 2 : The File1 and File2 are Zip files, hence extract them.

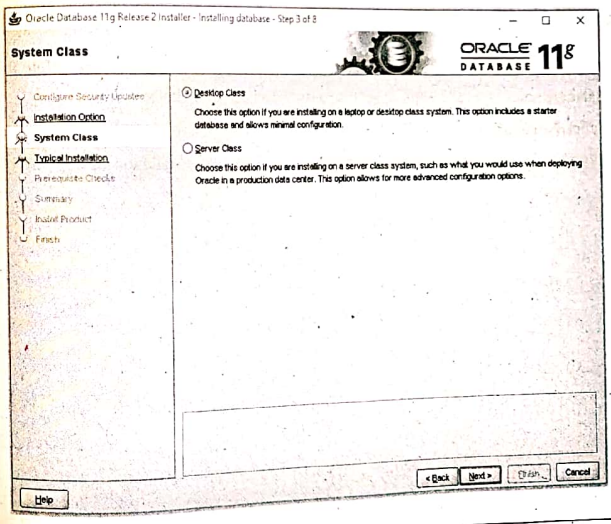
Step 3 : Open File2 folder, go to the folder **stage->Components**. Copy all the files and go to the File1 directory , Locate **stage** folder and open it copy the contents which you have copied.

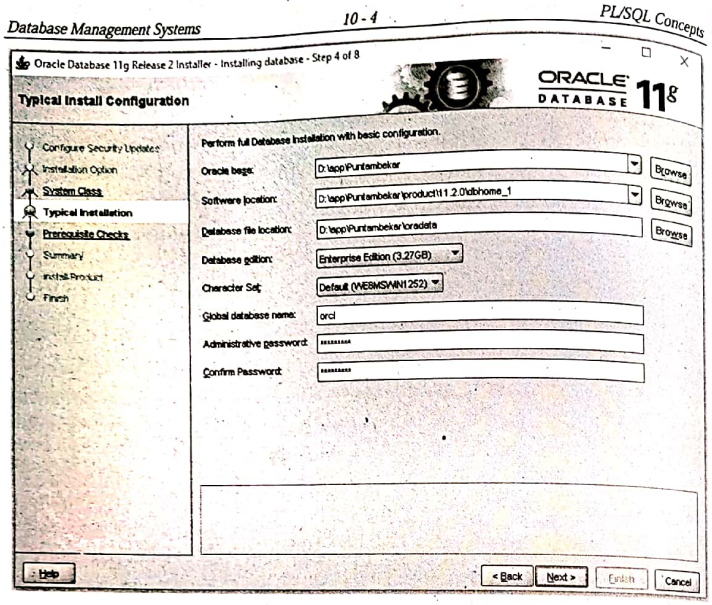
Step 4 : Open the folder for File1, locate setup.exe file. Double click it to start installation. Go on selecting the Next button to proceed the installation.

Step 5 : Select the installation option as follows –



Then Select **Desktop Class** installation. Click **Next button.**

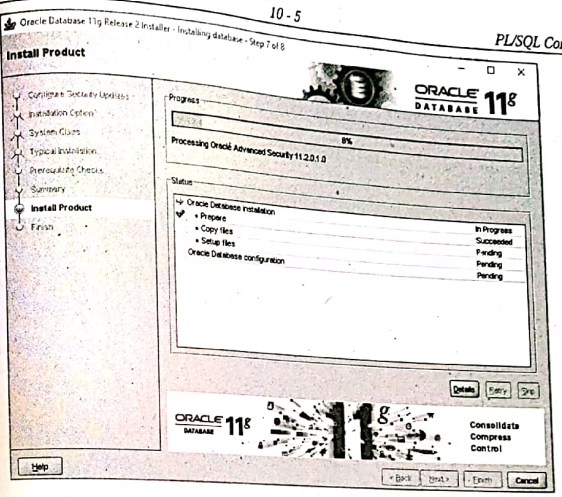




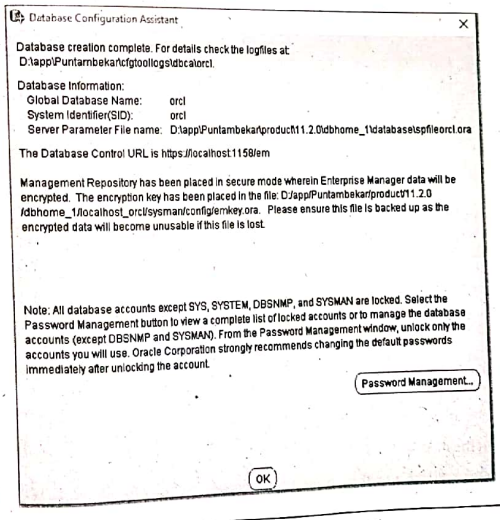
At this stage, you have to set administrative password for global database name. Note that the global database name is 'orcl'. You can set the password as per your choice.

Just click **Next** button.

Finally just click Finish button. It will then proceed for actual installation of copying of files step.



Step 6 : Now following window will appear. Here you have to set the password for the databases by clicking the **Password Management** button.



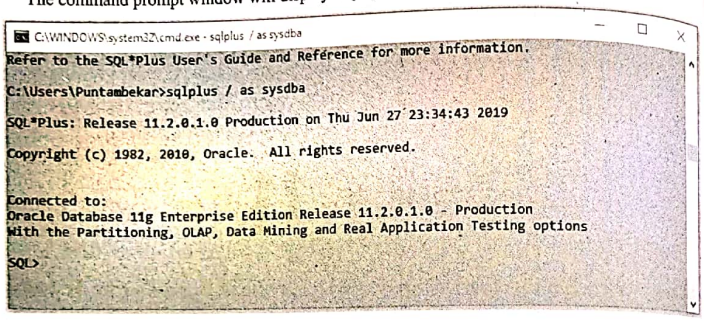
Scroll in the list that appears, Set the password for SYS and SYSTEM databases. I have set the passwords as "oracle". You are free to set any desired password of your choice.

Then click OK button.

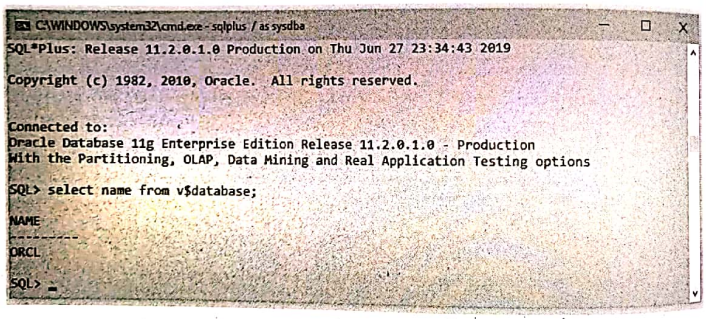
Open command prompt window and issue the following commands at the prompt

```
Set ORACLE-SID=orcl
sqlplus /as sysdba
```

The command prompt window will display SQL prompt as follows -



Step 7 : You can test the Oracle installation by typing the SELECT query as follows

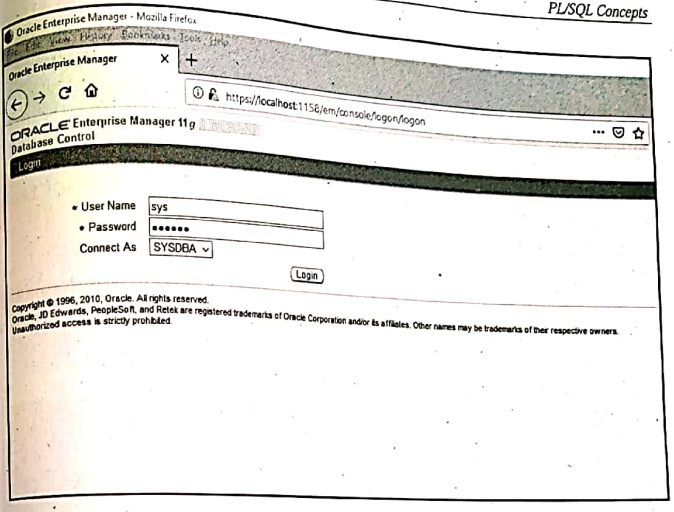


Then type quit and then exit at the SQL prompt to close this command prompt window. We can also check the installation is correct or not at GUI level as follows –

Just open the Web browser and type the URL as

https://localhost:1158/em

Following type of web page appears in your browser.

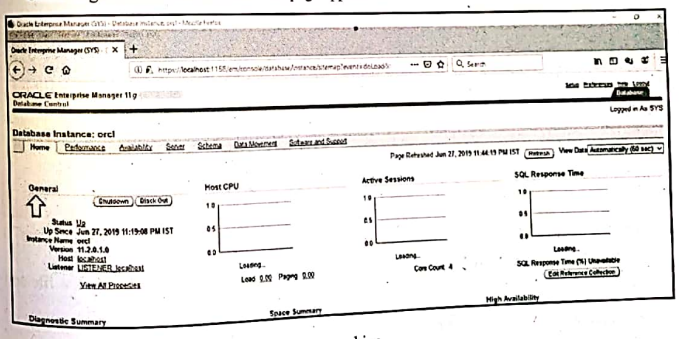


Username as **sys**

Password as **oracle**(you can choose any password of your choice)

Connect as SYSDBA

Click Login button and then the Web page appears as follows -



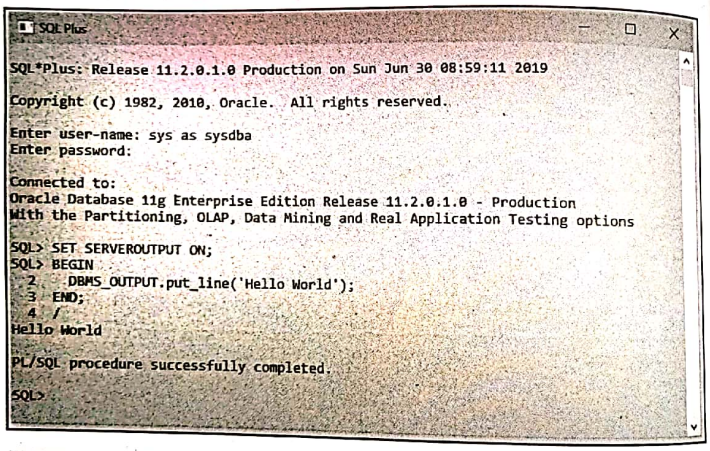
That it!!. Your Oracle is installed on your machine.

Troubleshooting : Sometimes at the start of installation itself, it give error about CREATE file on Command prompt, to resolve this error, just disable your Antivirus Software during the installation process.

10.3 Writing First PL/SQL Script

(1) Writing and Executing the PL/SQL script on SQL PLUS

- SQL Plus, which is a command-line interface for executing SQL statement and PL/SQL blocks provided by Oracle Database.
- Open SQL PLUS command prompt.
- Type user name and password. I am using the user name: sys as sysdba and password as oracle(This is the password which I set during installation. Refer installation procedure described in section 10.2, Step 7).
- Then the SQL prompt appears. Now we can execute our PL/SQL code from here.

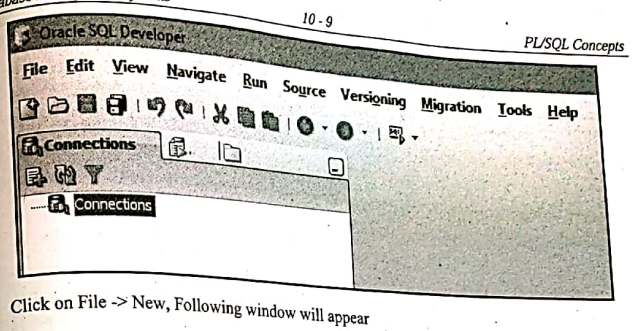


(2) Writing the PL/SQL script on Oracle SQL Developer

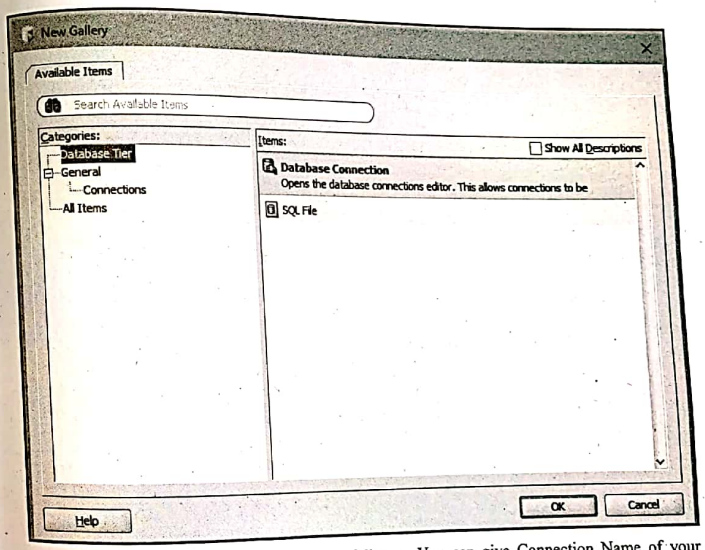
You can write PL/SQL script at the SQL Prompt or you can store the script in separate file and execute it.

For storing the script in separate file we need to use **Oracle SQL Developer**.

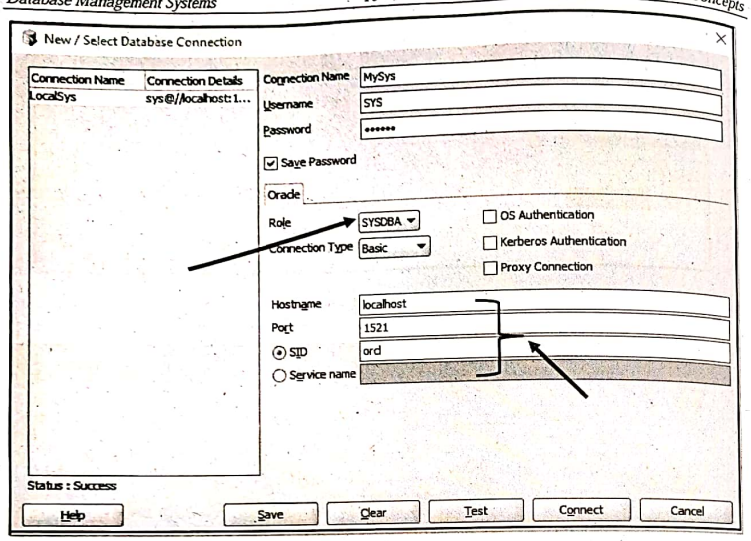
After installation of Oracle Package, this developer tool comes integrated with it. Click on it.



Click on File -> New, Following window will appear

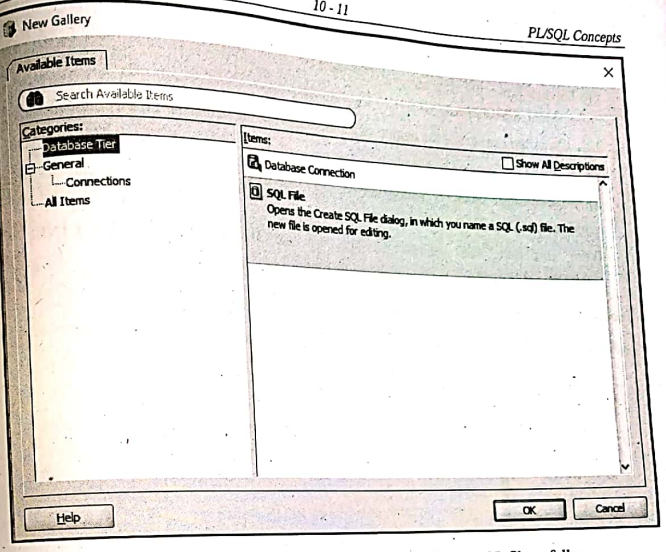


Now Enter user name and password as follows. You can give Connection Name of your choice. Then click Test button. On successful connection we get Success message. Then Click Connect button.

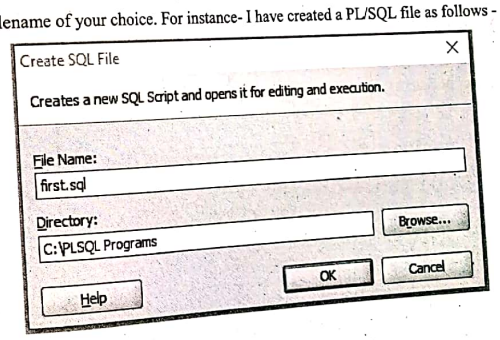


Now Click on **File->New.**

Following window appears. Click on **SQL File** and then Click **OK** button



Set the filename of your choice. For instance- I have created a PL/SQL file as follows -



The script is as follows –

```
SET SERVEROUTPUT ON;
BEGIN
 DBMS_OUTPUT.put_line('Hello World');
END;
/
```

Explanation of Script

(1) The first line of PL/SQL script is

```
SET SERVEROUTPUT ON;
```

This allows the user to see the output, when the script is executed on specified connection.

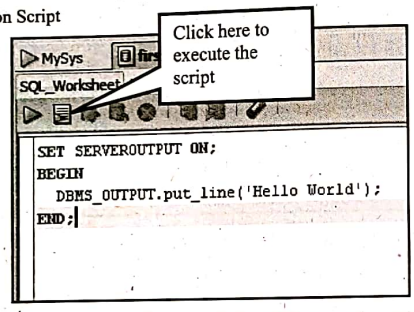
(2) Then **BEGIN** statement indicated the beginning of the execution block.

(3) The execution block is within the BEGIN and END statements. You can write any executable PL/SQL statements here.

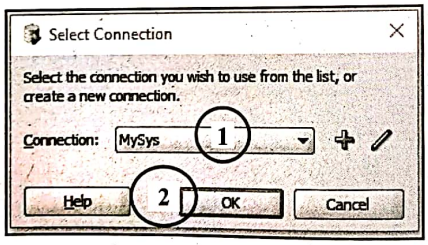
(4) For displaying the desired message we use **DBMS_OUTPUT.PUT_LINE**. The **DBMS_OUTPUT** is a built-in package that allows us to display output.

Execution of Script

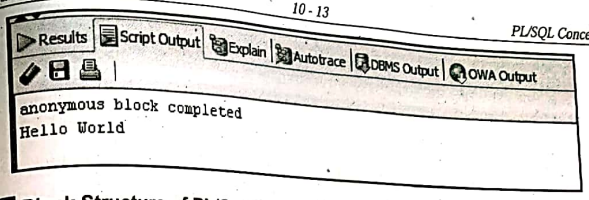
Click on Execution Script



The window for Select Connection will appear, Select the Connection and click OK button.



The output will be displayed at the bottom of the Editor window as follows



10.4 Block Structure of PL/SQL

- PL/SQL is a procedural language.
- The code is written in small blocks in PL/SQL.
- Broadly there are two types of blocks –

(1) Anonymous Block

- A block without a name is an anonymous block.
- An anonymous block is not saved in the Oracle Database server, so it is just for one-time use.
- PL/SQL anonymous blocks are useful for testing purposes

(2) Named Block

- A PL/SQL block has a name.
- A **Function or a Procedure** is an example of a named block.
- A named block is saved into the Oracle Database server first and then can be reused.

- A PL/SQL block consists of three sections: declaration, executable, and exception-handling sections. In a block, the executable section is mandatory while the declaration and exception-handling sections are optional. This structure of PL/SQL block is as follows

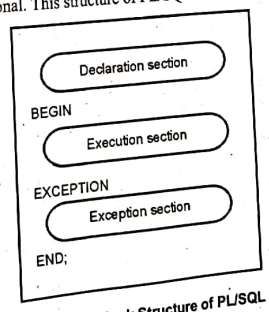


Fig. 10.4.1 Block Structure of PL/SQL

1) Declaration section

A PL/SQL block has a declaration section where you declare variables, allocate memory for cursors, and define data types.

2) Executable section

A PL/SQL block has an executable section. An executable section starts with the keyword **BEGIN** and ends with the keyword **END**. The executable section must have a least one executable statement, even if it is the NULL statement which does nothing.

3) Exception-handling section

A PL/SQL block has an exception-handling section that starts with the keyword EXCEPTION. The exception-handling section is where you catch and handle exceptions raised by the code in the execution section.

10.5 PL/SQL Data Types

Following are the types of PL/SQL data types

1. **Scalar :** These data types don't include any internal components. It includes data types such as NUMBER, DATE, BOOLEAN, etc.
2. **Large Objects (LOB) :** This type of data type stores objects that are relatively large in size and stored separately from other data types such as text, graphic images, video clips, sound, etc.
3. **Composite :** These type of data types have internal components that can be accessed individually. It includes records and collections.
4. **Reference :** As the name sounds, it includes pointers that refer to the location of the other data items.

Following are some commonly used data types in PL/SQL

Data Type	Description
Numeric	Numeric values on which arithmetic operations are performed. It includes sub types such as number, decimal, real, float, etc.
Character	Character values on which character operations such as strings are performed. It includes sub types such as char, varchar, varchar2, nvarchar2, etc.
Date and Time	This data type is for displaying the date and time values. The default date format is DD-MM-YY
Boolean	These include logical values on which logical operations are performed. The logical values are the Boolean values TRUE and FALSE and the value NULL.

Number	Number(Precision, Scale). **Precision** is the total number of digits and **scale** is the number of digits to the right of the decimal point. You cannot use constants or variables to specify precision and scale; you must use integer literals. To declare floating-point numbers, for which you cannot specify precision or scale because the decimal point can float to any position, use the following form without precision and scale : **NUMBER** The maximum precision that can be specified for a NUMBER value is 38 decimal digits. If you do not specify precision, it defaults to 39 or 40, or the maximum supported by your system, whichever is less. Scale, which can range from -84 to 127, determines where rounding occurs
Float	ANSI and IBM specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits).
Integer	ANSI and IBM specific integer type with maximum precision of 38 decimal digits
Real	Floating-point type with maximum precision of 63 binary digits (approximately 18 decimal digits).
Varchar2	Variable-length character string with maximum size of 32,767 bytes.
Rowid	Physical row identifier, the address of a row in an ordinary table.
Blob	This data types is used to store large binary objects in the database. Memory Capacity: 8 to 128 TB.
Clob	This data type is used to store large blocks of character data in the database. Memory Capacity: 8 to 128 TB.

10.6 PL/SQL Variables

- The variables in PL/SQL are declared in declaration section. The syntax of variable declaration is as follows -

Syntax

Variable-Name DataType(Precision/Dimension)

Example

```
Person_name varchar2(30);
```

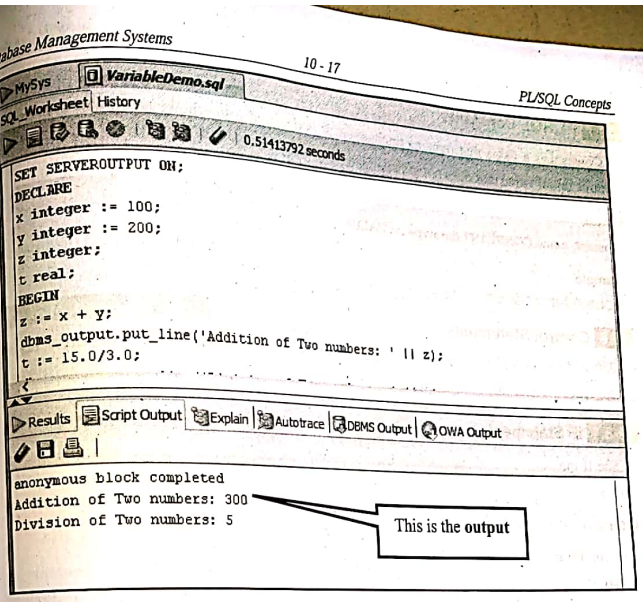
Here Person_name is a variable name and varchar2 is the data type with 30 as dimension.

- Constraints are associated with the variables defined in the code block. A constraint is a condition that is placed on the variable. Two commonly used constraints are –
  - **Constant** – This constraint will ensure that the value is not changed after a value is initially assigned to a variable. If a statement tries to change the variable value, an error will be displayed.
  - **Not Null** – This constraint will ensure that the variable always contains a value. If the statement attempts to assign an empty or a null value to that particular variable, the program will be error prone and will get abnormal termination of the program or the exception section will execute, if included in the program code.
  - **For Example –**
    PI constant number(5,2)=3.1415
- **Variable Initialization :** Variable can be initialized using :=. This sign includes a 'colon' with a succeeding 'equal to' sign. This particular sign assigns the parameter on the right hand side of the sign to the parameter or the variable on the left hand side of the sign.

Programming Example

```
SET SERVEROUTPUT ON;
DECLARE
x integer := 100;
y integer := 200;
z integer;
t real;
BEGIN
z := x + y;
dbms_output.put_line('Addition of Two numbers: ' || z);
t := 15.0/3.0;
dbms_output.put_line('Division of Two numbers: ' || t);
END;
/
```

Following Screenshot demonstrates above program



There are two types of variable scope.

1. Local Variables 2. Global Variables

For example

```
DECLARE
-- Global Variables
 a integer := 100;
 b integer := 200;
 dbms_output.put_line('a: ' || a);
 dbms_output.put_line('b: ' || b);
 BEGIN
 -- Local Variables
 c integer := 300;
 d integer := 400;
 dbms_output.put_line('c: ' || c);
 dbms_output.put_line('d: ' || d);
 END;
 /
```

10.7 PL/SQL Constants

Constant is a value that remains unchanged. It is declared using the keyword CONSTANT. It requires initial value that does not get changed.

Syntax

```
constant_name CONSTANT datatype := VALUE;
```

Example

```
PI CONSTANT NUMBER :=3.1415;
```

10.8 Control Statements

GTU : Winter-14, Marks 7

The control states determine the flow of execution of PL/SQL block. The control statements are of two types – conditional statements and loop statements.

10.8.1 IF Statement

The If-then-else type of statement is used to specify the condition in PL/SQL block. There are various ways by which the if statement can be written. Their syntaxes are as given below

(1) Syntax- IF-THEN

```
IF condition
THEN
Statement: {when condition is true, this statement executes}
END IF;
```

(2) Syntax- IF-THEN-ELSE

```
IF condition
THEN
  {statements to execute when condition is true}
ELSE
  {statements to execute when condition is False}
END IF;
```

(3) Syntax- If-THEN-ELSIF

```
IF condition#1
THEN
```



```
  {statements to execute when condition#1 is True}
ELSIF condition#2
THEN
  {statements to execute when condition#2 is True}
END IF;
```

(4) Syntax- If-THEN-ELSIF -ELSE

```
IF condition#1
THEN
  {statements to execute when condition#1 is True}
ELSIF condition#2
THEN
  {statements to execute when condition#2 is True}
ELSE
  {statements to execute when condition#1 and Condition#1 is False}
END IF;
```

Programming Example

```
SET SERVEROUTPUT ON;
DECLARE
 mark NUMBER :=50;
BEGIN
 dbms_output.put_line('Displaying Grades');
 IF( mark >= 70) THEN
  dbms_output.put_line('Grade A');
 ELSIF(mark >= 40 AND mark < 70) THEN
  dbms_output.put_line('Grade B');
 ELSIF(mark >=35 AND mark < 40) THEN
  dbms_output.put_line('Grade C');
 ELSE
  dbms_output.put_line('No Grade');
 END IF;
END;
/
```

Output(Using SQL PLUS)

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2    mark NUMBER :=50;
  3  BEGIN
  4    dbms_output.put_line('Displaying Grades');
  5    IF( mark >= 70) THEN
  6      dbms_output.put_line('Grade A');
  7    ELSIF(mark >= 40 AND mark < 70) THEN
  8      dbms_output.put_line('Grade B');
  9    ELSIF(mark >=35 AND mark < 40) THEN
 10      dbms_output.put_line('Grade C');
 11    ELSE
 12      dbms_output.put_line('No Grade');
 13    END IF;
 14  END;
 15  /
Displaying Grades
Grade B

PL/SQL procedure successfully completed.
```

Output

Output(Using Oracle SQL Developer)

```
anonymous block completed
Displaying Grades
Grade B
```

10.8.2 General Loop

- A General Loop in PL/SQL is used to execute a set of statements at least once before the termination of the loop.
- An EXIT condition has to be specified in the loop; otherwise the looping process will get into a never ending loop, also known as an Infinite Loop
- On encountering with EXIT statement, the loop exits.
- In a PL/SQL Loop, the statements are enclosed between the keywords LOOP and END LOOP.

- In every Loop, the statements are executed and then control restarts from the top of the loop until a certain condition is satisfied.

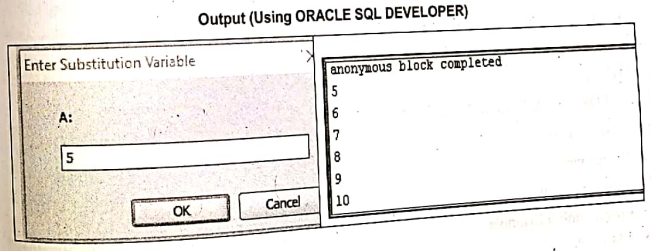
Syntax

```
LOOP
  Statement1;
  Statement2;
  EXIT;
END LOOP;
```

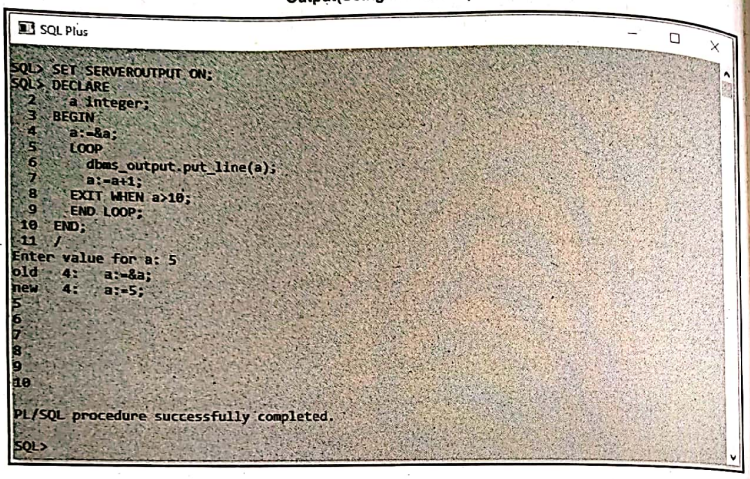
Programming Example

```
SET SERVEROUTPUT ON;
DECLARE
 a integer;
BEGIN
 a:=&a;
 LOOP
  dbms_output.put_line(a);
  a:=a+1;
 EXIT WHEN a>10;
 END LOOP;
END;
/
```

Output (Using ORACLE SQL DEVELOPER)



Output(Using SQL PLUS)



10.8.3 For Loop

- For loop executes sequence of statements for specific number of times
- There are starting and ending values in between which the statement executes.
- The counter is incremented by one each time.

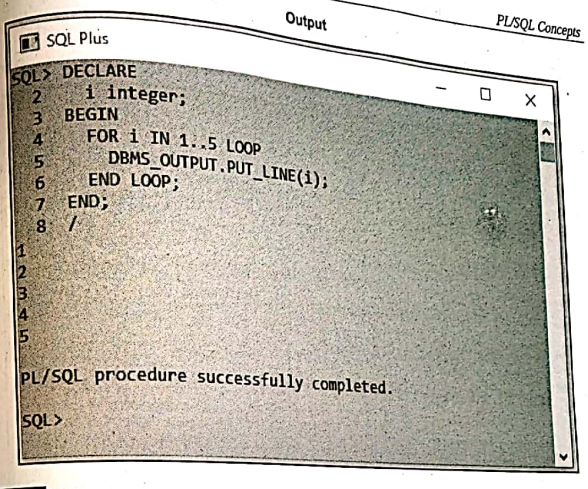
Syntax

```
FOR counter IN num1....num2 LOOP
statement1;
statement2;
END LOOP;
```

Programming Example

```
DECLARE
 i integer;
BEGIN
 FOR i IN 1..5 LOOP
  DBMS_OUTPUT.PUT_LINE(i);
 END LOOP;
END;
/
```


Output



10.8.4 While Loop

- This is a kind of loop that repeats the sequence of statements while given condition is true.
- Before entering the loop the condition is tested and if it is true then only the control executes the loop body. It executes until the condition becomes false.

Syntax

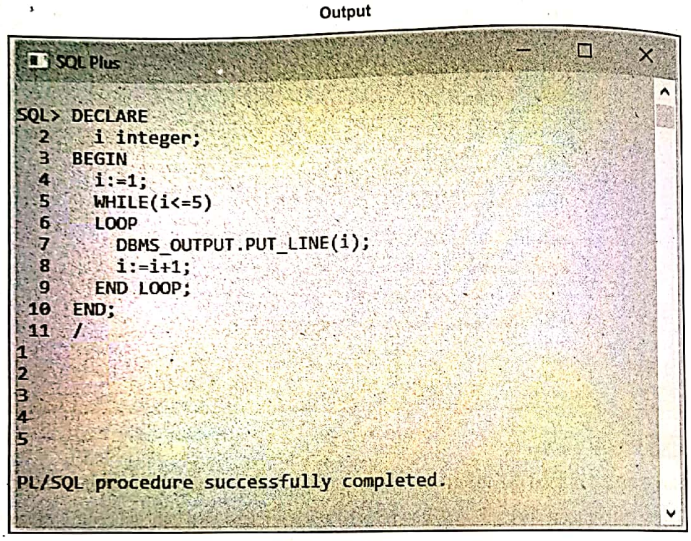
```
WHILE <condition>
LOOP statement1;
  statement2;
END LOOP;
```

Programming Example

```
DECLARE
 i integer;
BEGIN
 i:=1;
 WHILE(i<=5)
 LOOP
  DBMS_OUTPUT.PUT_LINE(i);
  i:=i+1;
```

```
 END LOOP;
END;
/
```

Output



10.8.5 CASE Statement

- The CASE statement is just similar to switch case statement in C.
- It allows you to execute the sequence of statements based on selector. The selector can be a variable, or a function or some expression.
- A CASE statement is evaluated from top to bottom. If it get the condition TRUE, then the corresponding THEN clause is executed and the execution goes to the END CASE clause.

Syntax

```
CASE [expression]
WHEN condition1 THEN statement1
  WHEN condition2 THEN statement2
  ...
  WHEN conditionn THEN statementn
```



```
  ELSE someStatement
  END
```

Programming Example

```
SET SERVEROUTPUT ON;
DECLARE
 grade CHAR(1) := 'A';
BEGIN
 CASE grade
WHEN 'A' THEN
 DBMS_OUTPUT.PUT_LINE('Distinction');
WHEN 'B' THEN
 DBMS_OUTPUT.PUT_LINE('First Class');
WHEN 'C' THEN
 DBMS_OUTPUT.PUT_LINE('Higher Second Class');
WHEN 'D' THEN
 DBMS_OUTPUT.PUT_LINE('Second Class');
WHEN 'F' THEN
 DBMS_OUTPUT.PUT_LINE('Pass');
 ELSE
  DBMS_OUTPUT.PUT_LINE('Fail');
 END CASE;
END;
/
```

Output

```
Distinction
```

10.8.6 Examples

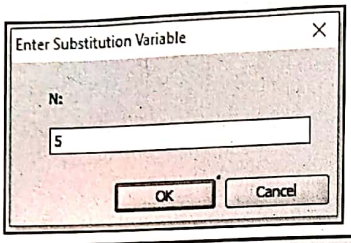
Example 10.8.1 *Write a PL/SQL block to print the factorial of number entered by the user.*

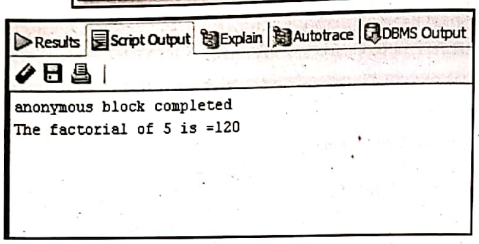
Solution :

```
SET SERVEROUTPUT ON;
DECLARE
    n NUMBER;
    f NUMBER:=1;
    i NUMBER;

BEGIN
    n:=&n;
    FOR i IN 1..n
    LOOP
     f:=f*i;
    END LOOP;
```

```
    DBMS_OUTPUT.PUT_LINE('The factorial of '||n||' is ='||f);
END;
/
```





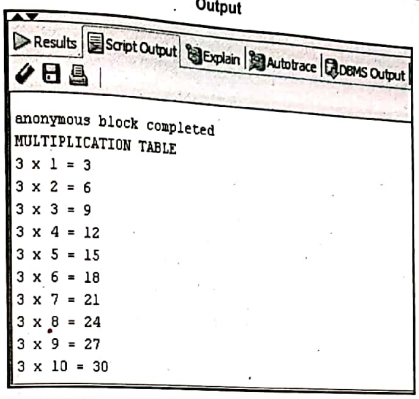
Example 10.8.2 *Write a PL/SQL block to display a multiplication table.*

Solution :

```
SET SERVEROUTPUT ON;
DECLARE
    n NUMBER;
    i NUMBER;

BEGIN
 DBMS_OUTPUT.PUT_LINE('MULTIPLICATION TABLE');
    n:=&n;
    FOR i IN 1..10
    LOOP
     DBMS_OUTPUT.PUT_LINE(n||' x '||i||' = '||n*i);
    END LOOP;
END;
/
```


Output

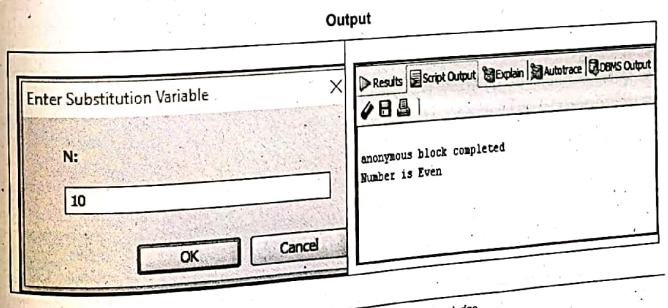


Example 10.8.3 *Write a PL/SQL block to check whether the number entered by the user is even or odd.*

Solution :

```
SET SERVEROUT ON;
DECLARE
    n NUMBER:=&n;
BEGIN
    IF MOD(n,2)=0 THEN
     DBMS_OUTPUT.PUT_LINE('Number is Even');
    ELSE
     DBMS_OUTPUT.PUT_LINE('Number is Odd');
    END IF;
END;
/
```

Output



Example 10.8.4 *Write a PL/SQL block to print the sum of even numbers from 1 to 100.*

GTU : Winter-14, Marks 7

Solution. :

```
DECLARE
 i integer;
 sum1 integer;
BEGIN
 i:=0;
 sum1 :=0;
 WHILE i<=100
 LOOP
   sum1:=sum1+i;
  i:=i+2;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE(sum1);
END;
/
```

Output

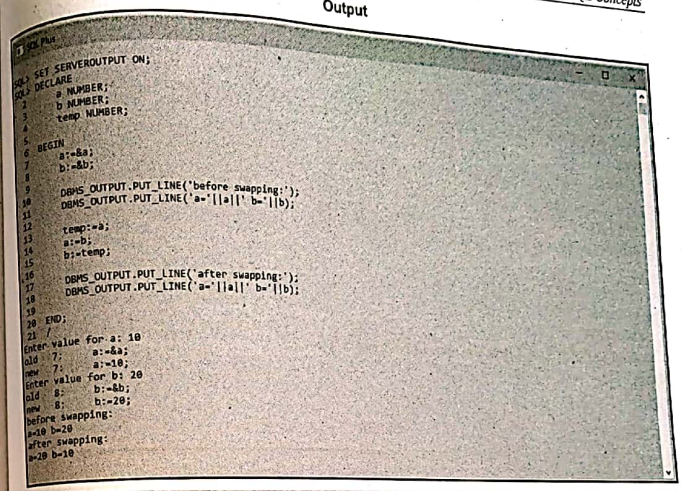
```
2550
```

Example 10.8.5 *Write a PL/SQL block to swap two numbers.*

Solution :

```
SET SERVEROUTPUT ON;
DECLARE
   a NUMBER;
   b NUMBER;
   temp NUMBER;
BEGIN
   a:=&a;
   b:=&b;
   DBMS_OUTPUT.PUT_LINE('before swapping:');
   DBMS_OUTPUT.PUT_LINE('a='||a||' b='||b);
   temp:=a;
   a:=b;
   b:=temp;
   DBMS_OUTPUT.PUT_LINE('after swapping:');
   DBMS_OUTPUT.PUT_LINE('a='||a||' b='||b);
END;
/
```


Output



```
SQL Plus
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2     a NUMBER;
  3     b NUMBER;
  4     temp NUMBER;
  5  BEGIN
  6     a:=&a;
  7     b:=&b;
  8
  9     DBMS_OUTPUT.PUT_LINE('before swapping:');
 10     DBMS_OUTPUT.PUT_LINE('a='||a||' b='||b);
 11
 12     temp:=a;
 13     a:=b;
 14     b:=temp;
 15
 16     DBMS_OUTPUT.PUT_LINE('after swapping:');
 17     DBMS_OUTPUT.PUT_LINE('a='||a||' b='||b);
 18
 19
 20  END;
 21  /
Enter value for a: 10
old   7:     a:=&a;
new   7:     a:=10;
Enter value for b: 20
old   8:     b:=&b;
new   8:     b:=20;
before swapping:
a=10 b=20
after swapping:
a=20 b=10
```

Example 10.8.6 *Write a PL/SQL block to print the sum of odd numbers from 1 to 100.*

GTU : Winter-14, Marks 7

Solution :

```
DECLARE
 i integer;
 sum1 integer;
BEGIN
 i:=1;
 sum1 :=0;
 WHILE i<=100
 LOOP
   sum1:=sum1+i;
  i:=i+2;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE(sum1);
END;
/
```

Output

```
2500
```

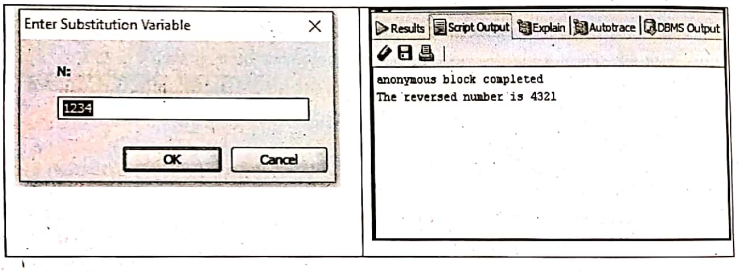
Example 10.8.7 *Write a PL/SQL block for reversing the given number.*

Solution :

```
DECLARE
    n NUMBER;
    i NUMBER;
    rev NUMBER:=0;
    r NUMBER;
BEGIN
    n:=&n;

    WHILE n>0
    LOOP
     r:=MOD(n,10);
     rev:=(rev*10)+r;
     n:=trunc(n/10);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('The reversed number is '||rev);
END;
/
```

Output



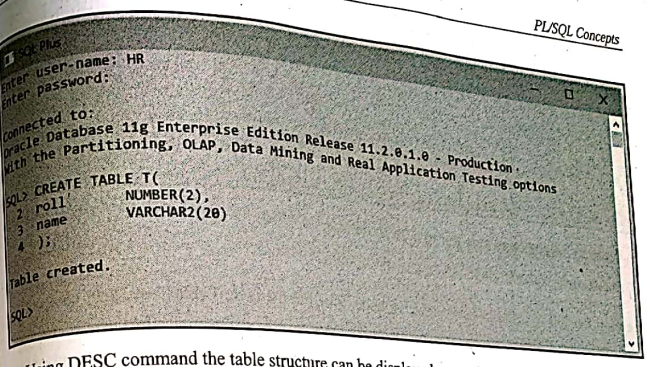
10.9 Handling Database Tables using PL/SQL

We can create a database table in Oracle database using two ways either using SQL prompt or using **Oracle SQL Developer**.

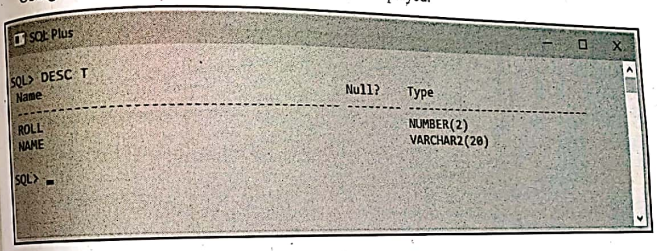
(1) Creating a Database Table Using SQL PLUS

Open SQL PLUS window, type the user-name and password. I am using HR schema under which the table will be created. The password which I have set during installation of ORACLE is HR. So by entering correct user name and password the SQL prompt will appear. Just give the CREATE table query for creating the table.

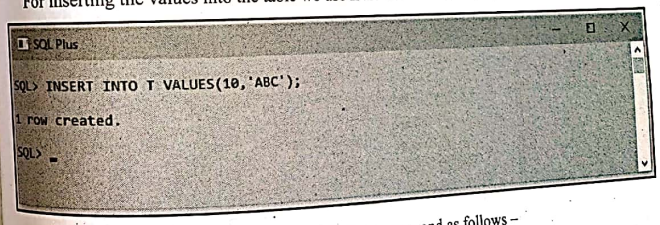
Following screenshot illustrates creation of table named T with two fields – **roll** and **name**.



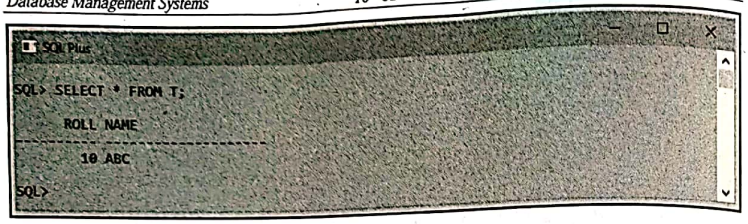
Using DESC command the table structure can be displayed.



For inserting the values into the table we use INSERT command as follows –



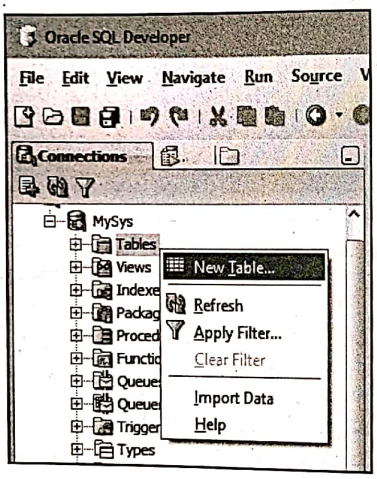
The table can then be displayed using SELECT command as follows –

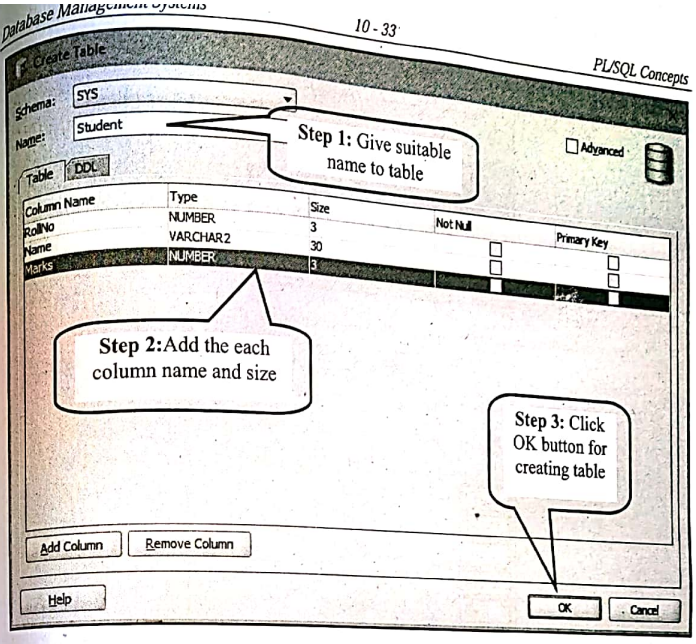


(2) Creating a Database Table Using SQL Developer

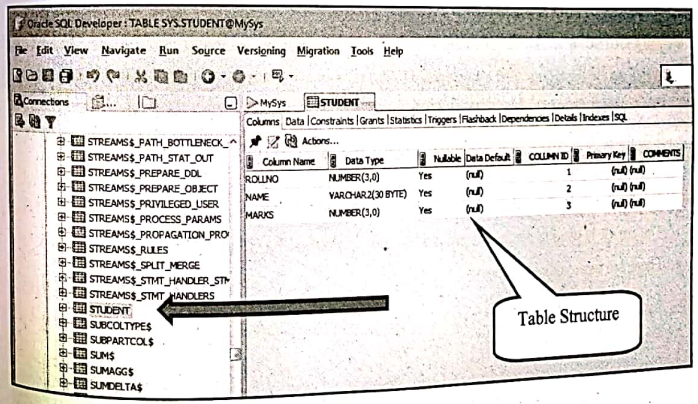
Open Oracle SQL Developer and follow these steps -

Step 1 : Expand your Database Connection node. Find out the node **Table**. Just Right click on it, click on New Table. Here I have already created a connection by name **MySys** (Refer section 10.3(2) to know how to create a connection), so I am creating a table under this schema. Here is the screenshot.

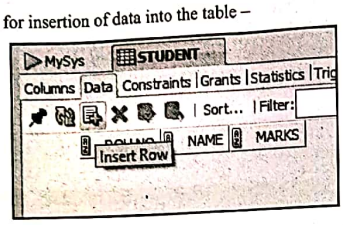




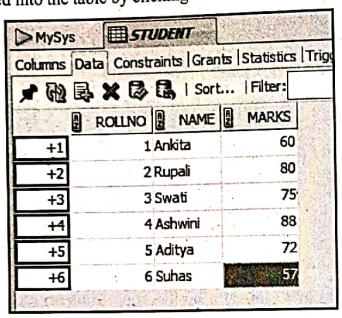
Now you can see your table structure by Selecting **Student** Table from Table node of your database Connection List. Just refer following screenshot



Just click on **Data** tab for insertion of data into the table –



The data can be inserted into the table by clicking to enter the value to each column –



Then click Save button. The data will be inserted into the table.

10.10 Cursors

GTU : Summer-17, Winter-15, 17, Marks 4

- When an SQL statement is processed, Oracle creates a memory area known as **context area**. A cursor is a **pointer to this context area**.
- It contains all information needed for processing the statement.
- In PL/SQL, the context area is controlled by Cursor.
- A cursor contains information on a SELECT statement and the rows of data accessed by it.
- The cursor is used to fetch and process the rows returned by SQL statement one at a time.
- There are two types of cursors –
  - Implicit cursor
  - Explicit cursor

(1) Implicit cursor

- Whenever Oracle executes an SQL statement such as SELECT INTO, INSERT, UPDATE, and DELETE, it creates an implicit cursor.
- The Implicit Cursor is the Default Cursor in PL/SQL block.
- Orcale provides some attributes known as Implicit cursor's attributes to check the status of DML operations. Some of them are as follows –
  - **%ROWCOUNT :** It Returns the number of rows influenced by an UPDATE, DELETE or an INSERT statement.
  - **%NOTFOUND :** It Returns TRUE if an UPDATE, DELETE or an INSERT statement influenced zero rows or a SELECT INTO statement returned no rows. Else, it Returns a FALSE.
  - **%FOUND :** It Returns TRUE if an UPDATE, DELETE or an INSERT statement influenced one or more rows or a SELECT INTO statement returned one or more rows. Else, it returns FALSE.
  - **%ISOPEN :** It Returns FALSE for Implicit cursors as Oracle closes the PL/SQL cursor by default after processing its associated PL/SQL statement.

Programming Example : Following is an example program that uses attributes of implicit cursor to demonstrate number of rows affected when SELECT query is executed for Student table

```
SET SERVEROUTPUT ON;
DECLARE
 s_name student.name%type;
BEGIN
 SELECT name INTO s_name
 FROM student
 WHERE marks>=70;
 DBMS_OUTPUT.PUT_LINE('Number of rows processed: '||sql%rowcount);
END;
/
```

Output

```
Number of rows processed:5
```

(2) Explicit cursor

- Explicit cursors are used when you are executing a SELECT statement query that will return more than one row.
- Cursors can process one record at a given point of time even though it stores more than one record.

- An explicit cursor is defined in the declaration section of the PL/SQL Block.
- Explicit cursors are used if you need to have better control over the Context Area via Cursor.

Syntax for Declaration of Explicit Cursor

```
CURSOR cusrsor_name
IS
SELECT statement
```

For example

```
CURSOR MyCursor
IS
SELECT * FROM Student;
```

The explicit cursor works in four stages –

1) **Declaration of cursor :** The declaration of cursor is in declaration section of PL/SQL block. The name of the cursor requires to be defined along with the SELECT Statement.

Syntax

```
CURSOR cursor-name IS
SELECT statement;
```

2) **Open the cursor :** Opening the Cursor also means allocating the memory for the Cursor in the Context Area which thereby makes it sufficient to fetch and store records in it.

Syntax

```
OPEN cursor-name;
```

3) **Fetch the cursor :** Fetching the Cursor involves retrieval of data using the Fetch statement. It is used to help the Cursor process and access records or rows at a time.

Syntax

```
FETCH cursor_name INTO variable_list;
```

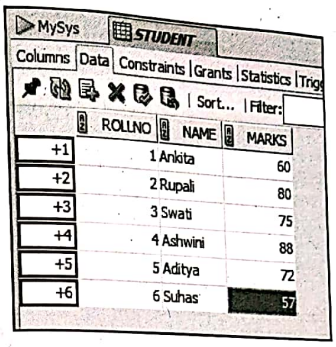
4) **Close the cursor :** Once the work associated for a Cursor to be completed is accomplished, it is necessary to release the Allocated Memory of the Cursor to let other tasks occupy memory.

Syntax

```
Close cursor_name;
```

Now let us understand how to write a simple explicit cursor and execute it.

Step 1 : Create a database table for using cursor. I have already created a Student table in section 10.9 (2). It is as follows



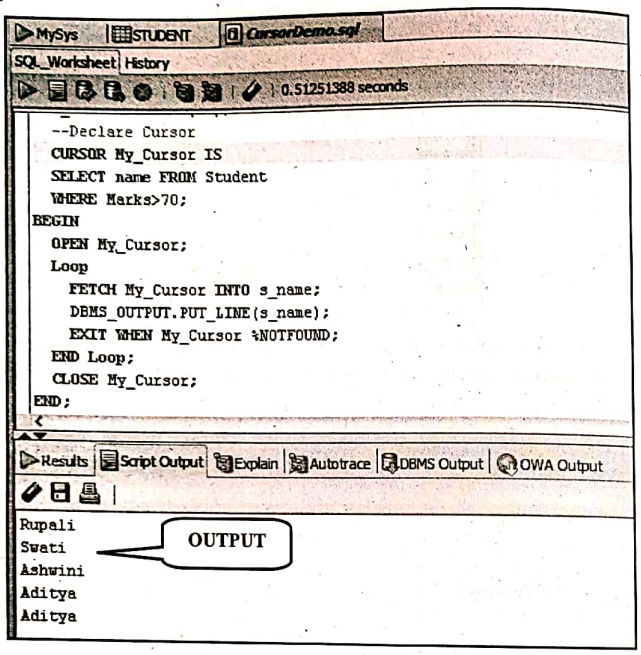
Step 2 : Under your current Connection type(In my case it is **MySys**), Open **File->New File.** Give suitable name to your file which stores the Cursor program. I have given the name as **CursorDemo.sql**

Step 3 : Write the program as follows

```
SET SERVEROUTPUT ON;
DECLARE
 s_name VARCHAR2(30);
 --Declare Cursor
 CURSOR My_Cursor IS
 SELECT name FROM Student
 WHERE Marks>70;
BEGIN
 OPEN My_Cursor;
 Loop
  FETCH My_Cursor INTO s_name;
  DBMS_OUTPUT.PUT_LINE(s_name);
  EXIT WHEN My_Cursor %NOTFOUND;
 END Loop;
 CLOSE My_Cursor;
END;
```

Step 4 : Save your file and execute script by either hitting F5 or by **Run Script**

The output will be obtained as a list of names of those students who have marks>70. Here is the output



Program Explanation : In above code,

- The cursor is created to display the names of students who have scored more than 70 marks.
- The name of the cursor is **My_Cursor**. The SELECT query associated with it is
  SELECT name FROM STUDENT WHERE Marks >70;
- Then we open the My_Cursor.
- Inside the LOOP, we fetch students' name from Student table into variable **s_name** and each time the **s_name** is displayed. This loop is exited when no record is found. For that purpose the attribute %NOTFOUND is used.
- Finally the cursor is closed.

Example 10.10.1 *Write a PL/SQL block using explicit cursor that will display the customer name, the fixed deposit number and the fixed deposit amount of the first 5 customers holding the highest amount in fixed deposits.*
Use the following database:
cust_mstr(custno,name, occupation)
fd_dtls(fd_ser_no,fd_no,type,period,opndt,duedt,amt,dueamt)
acct_fd_cust_dtls(acct_fd_no,custno)

GTU : Summer-17, Marks 4

Solution :

```
DECLARE My_Cur IS
SELECT name, fd_no,amt
FROM cust_mstr C, fd_dtls F,acct_fd_cust_dtls A
WHERE C.custno=A.custNo AND F.fd_no=A.acct_fd_no
ORDER BY amt DESC;

s_name   VARCHAR2(30);
s_fd_no  fd_dtls.fd_no%TYPE;
s_amt    fd_dtls.amt%TYPE;
BEGIN
OPEN My_Cur;
   LOOP FETCH My_Cur INTO s_name,s_fd_no,s_amt;
      DBMS_OUTPUT.PUT_LINE(s_name ||'  '||s_fd_no|| ' '||s_amt);
EXIT WHEN (My_Cur%ROWCOUNT-1)=5 OR  My_Cur %NOTFOUND;
END LOOP;
   CLOSE My_Cur;
END;
```

Example 10.10.2 *Write a PL/SQL cursor to display the names and branch of all students from the STUDENT relation.*

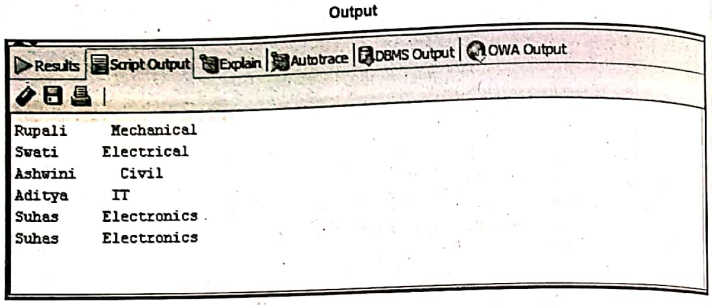
GTU : Winter-15, Marks 4

Solution :

```
SET SERVEROUTPUT ON;
DECLARE
 s_name VARCHAR2(30);
 s_branch VARCHAR2(20);
 --Declare Cursor
 CURSOR My_Cursor IS
 SELECT name,Branch FROM Student;
BEGIN
 OPEN My_Cursor;
 Loop
  FETCH My_Cursor INTO s_name,s_branch;
  DBMS_OUTPUT.PUT_LINE(s_name||'   '||s_branch);
  EXIT WHEN My_Cursor %NOTFOUND;
 END Loop;
```

```
CLOSE My_Cursor;
END;
```

Output



Review Question

1. *Explain Cursor in PL/SQL.* **GTU : Winter-17, Marks 3**

10.11 Stored Procedures

GTU : Summer-18, Winter-18, Marks 7

- Stored procedure is a type of subprogram in PL/SQL block. It is a group of statements that can be called by its name.
- This is a subprogram that does **not return a value directly**.
- A procedure is created with the CREATE OR REPLACE PROCEDURE statement

Syntax

```
CREATE or REPLACE Procedure Procedure_Name
[(Parameter_Name [IN | OUT | IN OUT ] Type [...])]
[IS | AS]
BEGIN
    Procedure_Body
END;
```

- Where
  - **Procedure_Name** is used to specify the name of the procedure.
  - **CREATE** keyword is used to develop a new procedure and [OR REPLACE] option allows us to modify an existing procedure.
  - The optional **parameter list** contains Name and Types of Parameters.

- There are three **ways to pass parameters in procedures -**
  - **IN** represents that argument value will be passed from outside the Procedure. It is a read-only parameter. Parameters are passed by reference. Inside the procedure or a sub-program, an IN Parameter acts as a constant. It cannot be assigned a value. It is the default mode of parameter passing.
  - An **OUT** parameter returns a value to the calling program. OUT represents that this parameter will be used to return a value outside of the procedure.
  - An **IN OUT** parameter passes an initial value to the sub-program and returns an updated value to the caller. We can read and write values using this parameter.
- The Executable part is included in the Procedure Body.
- We can create a procedure without passing the parameters or by passing the parameters.
- WE can execute the procedure using **Execute** keyword.
- A standalone procedure can be called by using the EXECUTE keyword or by calling or mentioning the procedure by its name from a PL/SQL block.

Syntax

```
Execute [Procedure-Name];
```

10.11.1 Procedures without Parameter

Let us write a simple procedure for displaying some welcome message.

Step 1 : Create a simple sql file that contains a procedure

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE MyMessage ← Creation of procedure
IS
BEGIN
  dbms_output.put_line('Welcome friend!!!');
END;
/
EXECUTE MyMessage; ← This is for executing the procedure
```

Step 2 : Save above output and compile the code. The output will be as follows –

```
PROCEDURE MyMessage Compiled.
anonymous block completed
Welcome friend!!!
```

Another way of execution of procedure

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE MyMessage ← Creation of procedure
IS
BEGIN
  dbms_output.put_line('Welcome friend!!!');
END;
/
BEGIN ← This is for calling the procedure
  MyMessage;
END;
/
```

Save and compiler above code to get the output.

10.11.2 Procedures with Parameters

Following example show the procedure for addition of two numbers. The result is stored in third variable which is an output variable.

Step 1 : We will create a procedure and store it in an sql file as follows

```
CREATE OR REPLACE PROCEDURE AddProc(x IN NUMBER, y IN NUMBER, z OUT NUMBER)
IS
BEGIN
  z:=x+y;
END;
/
```

Two Input and One output parameter

Save and compile above code.

Step 2 : The call to the procedure can be made from another file. The code for calling the procedure for execution is as follows –

```
SET SERVEROUTPUT ON;
DECLARE
a NUMBER;
b NUMBER;
c NUMBER;

BEGIN
  a := 10;
  b := 20;
  AddProc(a,b,c);
  dbms_output.put_line('Addition of two numbers:'||c);
END;
```

Save and compile above code

Step 3 : The output will be as follows –

```
anonymous block completed
Addition of two numbers:30
```

10.11.3 Stored Procedure for Handling Database Table

Using a stored procedure we can access the database table and can perform insertion, deletion and updation of data.

Following is a simple example of procedure using which we insert some record in Student table.

Step 1 : Write a procedure as follows

```
CREATE OR REPLACE PROCEDURE InsertStudent(roll IN NUMBER,
name IN VARCHAR2,
marks IN NUMBER,
branch IN VARCHAR2)
IS
BEGIN
INSERT INTO student VALUES(roll,name,marks,branch);
END;
/
```

Save and compile the above code.

Step 2 : Now call the procedure. The code is as follows

```
SET SERVEROUTPUT ON;
BEGIN
  InsertStudent(7,'Akash',78,'CIVIL');
  DBMS_OUTPUT.PUT_LINE('One row inserted in Student Table!!!');
END;
/
```

Save and compile the above code.

Step 3 : The output can be as follows

```
anonymous block completed
One row inserted in Student Table!!!
```

Step 4 : You can cross-verify the above execution by opening the student table. It should display new row inserted into it.

	ROLLNO	NAME	MARKS	BRANCH
1	1	Ankita	60	CS
2	2	Rupali	80	Mechanical
3	3	Swati	75	Electrical
4	4	Ashwini	88	Civil
5	5	Aditya	72	IT
6	6	Suhas	57	Electronics
7	7	Akash	78	CIVIL

Inserted new row

Review Question

1. *Explain stored procedure with proper example* **GTU : Summer-18, Winter-18, Marks 7**

10.12 Stored Functions

- Stored function is a named block or subprogram in PL/SQL.
- In PL/SQL, a function takes one or more parameter and returns one value.
- The main difference between a PL/SQL function and a PL/SQL procedure is that a function returns the value while a procedure does not.
- The syntax for a function in PL/SQL is as follows:

```
CREATE [Or REPLACE] Function Function_Name
[(Parameter,..)]
Return Datatype
[IS | AS]
[Declaration Section]
BEGIN
[Executable Section]
END Function_Name;
```

Let us now understand how to write a function in PL/SQL with the help of example

Step 1 : Open New file in Oracle SQL Developer and write following code for creating a simple function for addition of two numbers.

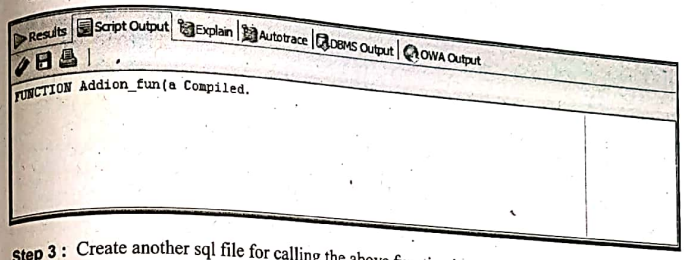
AdditionFunction.sql

```
CREATE OR REPLACE FUNCTION Addion_fun(a IN NUMBER, b IN NUMBER)
RETURN NUMBER IS
c NUMBER;
```



```
BEGIN
 c := a+b;
 RETURN c;
END;
/
```

Save this file , and compile it.



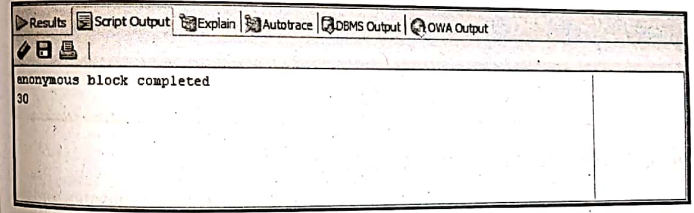
Step 3 : Create another sql file for calling the above function in an anonymous block.

CallAdditionFun.sql

```
SET SERVEROUTPUT ON;
BEGIN
 DBMS_OUTPUT.PUT_LINE(addion_Fun(10,20));
END;
/
```

Save above code and compile it.

Step 4 : On execution of the above code you get the output as



Example 10.12.1 *Write a PL/SQL Stored function to find maximum of two numbers.*

Solution :

Step 1 : The PL/SQL block that creates a function is as follows –

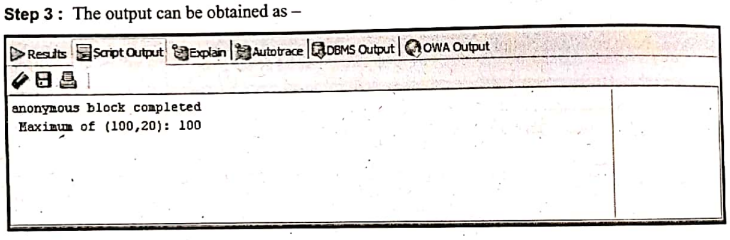
```
CREATE OR REPLACE FUNCTION Max_Fun(a IN number, b IN number)
RETURN number
IS
```

```
  c number;
BEGIN
  IF a > b THEN
    c:= a;
  ELSE
   'c:= b;
  END IF;
  RETURN c;
END;
/
```

Step 2 : The PL/SQL block that calls the above created function is as follows –

```
SET SERVEROUTPUT ON;
DECLARE
  a NUMBER := 100;
  b NUMBER := 20;
  c NUMBER;
BEGIN
  c := Max_Fun(a, b);
  DBMS_OUTPUT.PUT_LINE(' Maximum of (100,20): ' || c);
END;
/
```

Step 3 : The output can be obtained as –



10.12.1 PL SQL Stored Function For Table

We can write PL/SQL stored Function can be used to perform some functionality on the database Tables. Following example shows that we can write a function to find total number of students from **STUDENT** table. Here is a stepwise illustration –

Step 1 : Create a Student Table as follows –

ROLLNO	NAME	MARKS	BRANCH
1	Ankita	60	CS
2	Rupali	80	Mechanical
3	Swati	75	Electrical
4	Ashwini	88	Civil
5	Aditya	72	IT
6	Suhas	57	Electronics

Step 2 : Now create an sql file and create a function in it.

DisplayTotalFun.sql

```
CREATE OR REPLACE FUNCTION Total_Students
RETURN NUMBER IS
 t NUMBER(2) := 0;
BEGIN
  SELECT COUNT(*) INTO t
  FROM student;
  RETURN t;
END;
/
```

Save and compile above program

```
FUNCTION Total_Students Compiled.
```

Step 3 : Now we will write sql file for calling the above function

CallFun.sql

```
SET SERVEROUTPUT ON;
DECLARE
 t NUMBER(2);
BEGIN
  t := Total_Students();
  DBMS_OUTPUT.PUT_LINE('Total Number of Students in Table Student ' || t);
END;
/
```

Save above code and compile it.

```
anonymous block completed
Total Number of Students in Table Student 6
```

Example 10.12.2 *Write a PL/SQL function that returns the name of the student whose roll number is given by user.*

Solution :

Step 1 : Create a Student table. The sample table is as follows –

ROLLNO	NAME	MARKS	BRANCH
1	Ankita	60	CS
2	Rupali	80	Mechanical
3	Swati	75	Electrical
4	Ashwini	88	Civil
5	Aditya	72	IT
6	Suhas	57	Electronics

Step 2 : Write a function for retrieving the name of the student from the STUDENT table

```
CREATE OR REPLACE FUNCTION Display(roll NUMBER)
RETURN VARCHAR2 IS
  s_name VARCHAR2(20);
BEGIN
  SELECT name INTO s_name
  FROM student
  WHERE ROLLNO=roll;
   RETURN s_name;
END;
/
```

Save and compile above code.

Step 3 : now call the above function

```
SET SERVEROUTPUT ON;
DECLARE
  s_name VARCHAR2(20);
BEGIN
  s_name := Display(2);
  DBMS_OUTPUT.PUT_LINE('Name Student with Rollno 2 is: ' || s_name);
END;
/
```


Save and comile above file. You will get following output

```
anonymous block completed
Name Student with Rollno 2 is: Rupali
```

10.13 Database Triggers

GTU : Summer-18, Winter-14, 17, 18, Marks 4

- Trigger is something that is invoked automatically when some event occurs.
- PL/SQL triggers are **block structures or pre-defined programs**, which may be in-built or even explicitly developed by the programmers for a particular task.
- Trigger is **stored into database** and invoked repeatedly, when specific condition match.
- Triggers are stored programs, which are automatically executed or fired when some event occurs.
- Triggers are associated with response-based events such as a
  - Database Definition Language (DDL) statement such as CREATE, DROP or ALTER .
  - Database Manipulation Language (DML) statement such as UPDATE, INSERT or DELETE
  - Any other database operation such as a Startup, Shutdown, Logging in and Logging Out.

Syntax

```
CREATE OR REPLACE TRIGGER Trigger_Name
BEFORE or AFTER or INSTEAD OF
INSERT or UPDATE or DELETE
of Column_Name
ON Table_Name
[REFERENCING OLD AS O New AS N]
FOR EACH ROW
WHEN (Condition)
DECLARE
    Declaration Section
BEGIN
```

Execution Section

END;

Where

- CREATE [OR REPLACE] TRIGGER trigger_name: It creates or replaces an existing trigger with some trigger_name.
- {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
- [OF col_name]: This specifies the column name that would be updated.
- [ON table_name]: This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
- [FOR EACH ROW]: This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition): This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Advantages and need of trigger

1) Triggers can be used for implementing **referential integrity constraints**.
2) By using triggers, business rules and transactions are **easy to store** in database and can be used consistently even if there are future updates to the database.
3) It **controls updates** allowed in a database.
4) When a change happens in a database a trigger **can adjust the change** to the entire database.
5) Triggers are used for **calling stored procedures**.

Programming Example

Let us now see how to use trigger with the help of some example -

Example 10.13.1 *Write a trigger for insertion of a row into a person table. On firing the trigger name of the user who performed insertion operation should be displayed.*

Solution :

```
CREATE TABLE PersonTab (
 pname VARCHAR2(20)
);
```

Here table named **PersonTab** is created


```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER MyTrigger
BEFORE INSERT ON PersonTab
FOR EACH ROW
ENABLE
DECLARE
 usr_name VARCHAR2(20);
BEGIN
 SELECT user INTO usr_name FROM dual;
 DBMS_OUTPUT.PUT_LINE('Inserted a new row by user: '||usr_name);
END;
/
```

Code for actual trigger

```
INSERT INTO PersonTab VALUES('Sharda');
```

SQL query on execution of which the trigger gets fired.

Output

```
*Cause:
*Action:
TRIGGER MyTrigger Compiled.
1 rows inserted
Inserted a new row by user: HR
```

You can cross-verify the insertion of data into the table by opening the corresponding table.

For instance -

PNAME
1Sharda

Review Questions

1. *Write a note on trigger.* **GTU : Summer-18, Winter-18, Marks 3**
2. *What are triggers ? Explain the advantages and the needs of triggers.* **GTU : Winter-14, 17, Marks 4**

10.14 Oral Questions and Answers

Q.1 What is PL/SQL ?

Ans. :

- PL/SQL stands for Procedural Language extensions to the Structured Query Language (SQL).
- PL/SQL is a combination of SQL along with the procedural features of programming languages.

- It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL.

Q.2 What is anonymous block ?

Ans. :

- A block without a name is an anonymous block.
- An anonymous block is not saved in the Oracle Database server, so it is just for one-time use.
- PL/SQL anonymous blocks are useful for testing purposes.

Q.3 What is named block ?

Ans. :

- A PL/SQL block has a name.
- A Function or a Procedure is an example of a named block.

Q.4 Enlist various sections of PL/SQL program.

Ans. : A PL/SQL block consists of three sections : declaration, executable and exception-handling sections. In a block, the executable section is mandatory while the declaration and exception-handling sections are optional.

Q.5 What is Cursor in PL/SQL ?

Ans. :

- When an SQL statement is processed, Oracle creates a memory area known as context area. A cursor is a pointer to this context area.
- It contains all information needed for processing the statement.
- In PL/SQL, the context area is controlled by Cursor.

Q.6 What are types of cursor ?

Ans. : There are two types of cursors – 1) Implicit cursor and 2) Explicit cursor

Q.7 What is implicit cursor ?

Ans. :

- Whenever Oracle executes an SQL statement such as SELECT INTO, INSERT, UPDATE, and DELETE, it creates an implicit cursor.
- The implicit cursor is the default cursor in PL/SQL block.

Q.8 What is explicit cursor ?

Ans. :

- Explicit cursors are used when you are executing a SELECT statement query that will return more than one row.
- Cursors can process one record at a given point of time even though it stores more than one record.

Q.9 What is stored procedure in PL/SQL ?

Ans. :

- Stored procedure is a type of subprogram in PL/SQL block. It is a group of statements that can be called by its name.
- This is a subprogram that does not return a value directly.
- A procedure is created with the CREATE OR REPLACE PROCEDURE statement

For example -

```
CREATE OR REPLACE PROCEDURE MyMessage
IS
BEGIN
  dbms_output.put_line('Welcome friend!!!');
END;
```

Q.10 What is stored function in PL/SQL ?

Ans. :

- Stored function is a named block or subprogram in PL/SQL.
- In PL/SQL, a function takes one or more parameter and returns one value.

For example -

```
CREATE OR REPLACE FUNCTION Addion_fun(a IN NUMBER, b IN NUMBER)
RETURN NUMBER IS
c NUMBER;
BEGIN
 c := a+b;
 RETURN c;
END;
```

Q.11 What is the difference between procedure and function in PL/SQL ?

Ans. : The main difference between a PL/SQL function and a PL/SQL procedure is that a function returns the value while a procedure does not.

Q.12 What is database triggers ?

Ans. :

- PL/SQL triggers are block structures or pre-defined programs, which may be in-built or even explicitly developed by the programmers for a particular task.
- Trigger is stored into database and invoked repeatedly, when specific condition match.

Q.13 Give some advantages of having triggers in database.

Ans. :

1) Triggers can be used for implementing referential integrity constraints.
2) By using triggers, business rules and transactions are easy to store in database and can be used consistently even if there are future updates to the database.
3) It controls updates allowed in a database.
4) When a change happens in a database a trigger can adjust the change to the entire database.

❑❑❑

SOLVED MODEL QUESTION PAPER

[As per New Question Paper Pattern]

Database Management Systems

Semester – III (CE/IT)

Time : $2\frac{1}{2}$ Hours] **[Total Marks : 70**

Instructions :

1. **Attempt all questions.**
2. **Make suitable assumptions whenever necessary.**
3. **Figure to the right indicates full marks.**

Q.1 **(a)** *What is DBMS ? Enlist applications of DBMS.* **(Refer section 1.1)** (3)

(b) *Explain data abstraction in detail.* **(Refer section 1.2)** (4)

(c) *Explain network model. Give advantages and disadvantages of it.* **(Refer section 2.3)** (7)

Q.2 **(a)** *What is relational algebra ? Define relational algebra operation cross product with example* **(Refer section 3.2)** (3)

(b) *What is the difference between open source DBMS and commercial DBMS.* **(Refer section 3.8)** (4)

(c) *Solve the queries for the following database using relational algebra*

branch (branch-name, branch-city, assets)
customer (customer-name, customer-street, customer-only)
account (account-number, branch-name, balance)
loan (loan-number, branch-name, amount)
depositor (customer-name, account-number)
borrower (customer-name, loan-number)

(1 Find all loans over $1200
(2) Find the loan number for each loan of an amount greater than $1200
(3) Find the names of all customers who have a loan, an account, or both, from the bank
(4) Find the names of all customers who have a loan and an account at bank.
(5) Find the names of all customers who have a loan at the Perryridge branch.
(6) Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.
(7) Find the names of all customers who have a loan and an account at the Perryridge branch. **(Refer example 3.2.7)** (7)

OR

(c) *(i) Compute the closure of R(A,B,C,D,E) with the following set of functional dependencies*
A->BC, CD->E, B->D, E->A
List the candidate keys of R **(Refer example 4.2.8)**
(ii) Consider Schema R = (A,B,C,G,H,I) and the set F of functional dependencies {A->B, A->C, CG->H, CG->I, B->H}. Use (F)+ Prove (AG)+->I. **(Refer example 4.2.9)** (7)

Q.3 **(a)** *Write a note on query evaluation plan.* **(Refer section 5.1)** (3)

(b) *Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies*
F = {{A, B}→ C, A→ {D, E}, B→ F, F→{G, H}, D→{I, J} }
1. What is the key for R ? Demonstrate it using the inference rules.
2. Decompose R into 2NF, then 3NF relations. **(Refer example 4.7.2)** (4)

(c) *What is the use of an index structure and explain the concept of ordered indices.* **(Refer section 6.1)** (7)

OR

Q.3 **(a)** *Give difference between B and B+ Trees.* **(Refer section 6.2.2)** (3)

(b) *Explain steps of query processing with the help of neat diagram.* **(Refer section 5.1)** (4)

(c) *What is normalization ? Why normalization process is needed ? Explain 1NF, 2NF, 3NF with example.* **(Refer section 4.7)** (7)

Q. 4 **(a)** *Explain the term – transaction* **(Refer section 7.1)** (3)

(b) *Explain various states of transaction with suitable diagram.* **(Refer section 7.3)** (4)

(c) *Consider the following two transactions and schedule (time goes from top to bottom). Is this schedule conflict-serializable ? Explain why or why not ?* **(Refer example 7.5.1)** (7)

T_1	T_2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	

OR

Q. 4 **(a)** *Discuss the following with relevant examples :* **(Refer example 7.3.1)** (3)
1. A read only transaction
2. A read write transaction
3. An aborted transaction

(b) *Enlist advantages and disadvantages of two phase locking* **(Refer section 7.7.3)** (4)

(c) *What is concurrency ? What are the three problems due to concurrency ? How the problems can be avoided, explain for one of the three problems.* **(Refer section 7.6)** (7)

Q.5 **(a)** *What is the difference between authentication and authorization ?* **(Refer section 8.2)** (3)

(b) *What is SQL injection ? Explain it with suitable example.* **(Refer section 8.5)** (4)

(c) *Explain aggregate functions of SQL with suitable example.* **(Refer section 9.2)** (7)

OR

Q.5 **(a)** *What is cursor? Explain its types briefly.* **(Refer section 10.10)** (3)

(b) *Write a PL/SQL cursor to display the names and branch of all students from the STUDENT relation.* **(Refer example 10.10.2)** (4)

(c) *Explain stored procedure with proper example.* **(Refer section 10.11)** (7)

❑❑❑