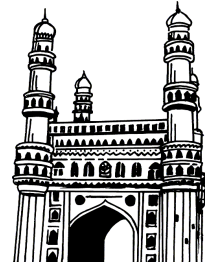


**Rahul's** ✓  
Topper's Voice

AS PER  
CBCS SYLLABUS



# B.Sc.

## II Year IV Sem

Latest 2021-22 Edition

# DATABASE MANAGEMENT SYSTEMS

- ☞ Study Manual
- ☞ Important Question
- ☞ Short Question & Answers
- ☞ Multiple Choice Questions
- ☞ Fill in the blanks
- ☞ Solved Model Papers

Syllabus covered for :

**Osmania University**  
**Kakatiya University**  
**Satavahana University**

Useful for :

**Mahatma Gandhi University**  
**Palamuru University**  
**Telangana University**

189/-



**Rahul Publications**™

Hyderabad. Ph : 66550071, 9391018098

All disputes are subjects to Hyderabad Jurisdiction only

# **B.Sc.**

## **II Year IV Sem**

# **DATABASE MANAGEMENT SYSTEMS**

*Inspite of many efforts taken to present this book without errors, some errors might have crept in. Therefore we do not take any legal responsibility for such errors and omissions. However, if they are brought to our notice, they will be corrected in the next edition.*

© No part of this publications should be reporduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording and/or otherwise without the prior written permission of the publisher

*Price ` . 189-00*

---

**Sole Distributors :**

**☎ : 65500071, Cell : 9391018098**

**VASU BOOK CENTRE**

**Shop No. 3, Beside Gokul Chat, Koti, Hyderabad.**

**Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.**

**Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.**

# **DATABASE MANAGEMENT SYSTEMS**

## **STUDY MANUAL**

<b>Important Questions</b>	<b>V - X</b>
<b>Unit - I</b>	<b>1 - 46</b>
<b>Unit - II</b>	<b>47 - 98</b>
<b>Unit - III</b>	<b>99 - 174</b>
<b>Unit - IV</b>	<b>175 - 226</b>

## **SOLVED MODEL PAPERS**

<b>Model Paper - I</b>	<b>227 - 227</b>
<b>Model Paper - II</b>	<b>228 - 228</b>
<b>Model Paper - III</b>	<b>229 - 229</b>

# SYLLABUS

## UNIT – I

**Introduction:** Database-System Applications, Purpose of Database Systems, View of Data, Database Languages, Relational Databases, Database Design, Data Storage and Querying, Transaction Management, Database Architecture, Database Users and Administrators.

**Introduction to the Relational Model:** Structure of Relational Databases, Database Schema, Keys, Schema Diagrams, Relational Query Languages, Relational Operations.

## UNIT – II

**Database Design and the E-R Model:** Overview of the Design Process, The Entity- Relationship Model, Constraints, Removing Redundant Attributes in Entity Sets, Entity-Relationship Diagrams, Reduction to Relational Schemas, Entity-Relationship Design Issues, Extended E-R Features, Alternative Notations for Modeling Data, Other Aspects of Database Design.

**Relational Database Design:** Features of Good Relational Designs, Atomic Domains and First Normal Form, Decomposition Using Functional Dependencies, Functional- Dependency Theory, Decomposition Using Multivalued Dependencies, Normal Forms-2 NF, 3 NF, BCNF, The Database Design Methodology for Relational Databases.

## UNIT – III

**Introduction to SQL:** Overview of the SQL Query Language, SQL Data Definition, Basic Structure of SQL Queries, Additional Basic Operations, Set Operations, Null Values, Aggregate Functions, Nested Subqueries, Modification of the Database.

**Intermediate SQL:** Join Expressions, Views, Transactions, Integrity Constraints, SQL Data Types and Schemas, Authorization.

**Advanced SQL:** Accessing SQL from a Programming Language, Functions and Procedures, Triggers, Recursive Queries.

## UNIT – IV

**Transaction Management:** Transaction Support–Properties of Transactions, Database Architecture, Concurrency Control–The Need for Concurrency Control, Serializability and Recoverability, Locking Methods, Deadlock, Time Stamping Methods, Multi-version Timestamp Ordering, Optimistic Techniques, Granularity of Data Items, Database Recovery–The Need for Recovery, Transactions and Recovery, Recovery Facilities, Recovery Techniques, Nested Transaction Model. Security: Database Security–Threats, Computer-Based Controls–Authorization, Access Controls, Views, Backup and Recovery, Integrity, Encryption, RAID.

# *Contents*

Topic	Page No.
<b>UNIT - I</b>	
1.1 Introduction to Database .....	1
1.2 System Applications .....	5
1.3 Purpose of Database Systems .....	5
1.4 View of data .....	9
1.5 Database Languages .....	11
1.6 Relational Databases .....	12
1.7 Database Design .....	15
1.8 Data storage and Querying .....	20
1.9 Transaction Management .....	21
1.10 Database Architecture .....	23
1.11 Database users and Administrators .....	24
1.12 Introduction to the Relational Model .....	26
1.13 Structure of Relational Databases .....	30
1.14 Database Schema .....	31
1.15 Keys .....	32
1.16 Schema Diagrams .....	34
1.17 Relational Query Languages and Relational Operations .....	35
➤ Short Question and Answers .....	44
➤ Choose the Correct Answers .....	45
➤ Fill in the blanks .....	46
<b>UNIT - II</b>	
2.1 Database design Model and ER – Model .....	47
2.1.1 Overview of design process .....	47
2.1.2 The Entity Relationship model .....	51
2.1.3 Constraints .....	55
2.1.4 Removing redundant attributes in entity set .....	56

Topic	Page No.
2.1.5 Entity Relationship Diagram .....	58
2.1.6 Reduction to relational schemas .....	62
2.1.7 Entity Relationship Design Issues .....	66
2.1.8 Extended ER Features .....	67
2.1.9 Alternative notations for modelling data .....	69
2.1.10 Other aspects of database design .....	70
2.2 Relational Database Design .....	71
2.2.1 Features of good relational designs .....	71
2.2.2 Atomic domains and first normal form .....	73
2.2.3 Decomposition using functional dependencies, functional dependency theory, Decomposition using multi valued dependencies .....	80
2.2.4 Normal forms - 2NF, 3NF and BCNF .....	84
2.2.5 The database design methodology for Relational databases .....	88
➤ Short Question and Answers .....	89
➤ Choose the Correct Answers .....	97
➤ Fill in the blanks .....	98

### UNIT - III

3.1 Introduction to SQL .....	99
3.1.1 Over Views of SQL .....	99
3.1.2 SQL Data definition .....	100
3.1.3 Basic structure of SQL Queries .....	120
3.1.4 Additional basic operations .....	122
3.1.5 Set Operations .....	127
3.1.6 Null values .....	129
3.1.7 Aggregate functions .....	131
3.1.8 Nested subqueries .....	133
3.1.9 Modification of the database .....	136
3.2 Intermediate SQL .....	138
3.2.1 Join Expressions .....	138
3.2.2 Views .....	145

Topic	Page No.
3.2.3 Transactions .....	147
3.2.4 Integrity constraints .....	150
3.2.5 SQL Data types and Schemas .....	154
3.2.6 Authorization .....	156
3.3 Advanced SQL .....	156
3.3.1 Accessing SQL from a programming language .....	156
3.3.2 Functions and procedures .....	158
3.3.3 Triggers .....	161
3.3.4 Recursive Queries .....	165
➤ Short Question and Answers .....	167
➤ Choose the Correct Answers .....	173
➤ Fill in the blanks .....	174

#### UNIT - IV

4.1 Transaction Management .....	175
4.1.1 Transaction Support, Properties of Transactions .....	175
4.1.2 Database Architecture .....	177
4.2 Concurrency Control .....	178
4.2.1 The Need for Concurrency Control .....	178
4.3 Serializability .....	183
4.4 Recoverability .....	187
4.5 Locking Methods .....	188
4.6 Deadlock Prevention and Detection .....	190
4.6.1 Timestamp Ordering, Optimistic Techniques, Granularity of Data Items .....	192
4.7 Database Recovery .....	194
4.7.1 The Need for Recovery .....	194
4.7.2 Transactions and Recovery, Recovery Facilities .....	196
4.7.3 Recovery Techniques .....	198
4.7.4 Nested Transaction Model .....	200

Topic	Page No.
4.8 Database Security .....	201
4.8.1 Threats to Database Security .....	202
4.8.2 Computer Based Controls .....	204
4.9 Authorization .....	206
4.10 Access Control Mechanisms .....	210
4.11 Views .....	211
4.11.1 Backup and Recovery, Integrity .....	213
4.12 Database Encryption and Decryption .....	214
4.13 Redundant Array of Independent Disks (RAID) .....	216
➤ Short Question and Answers .....	218
➤ Choose the Correct Answers .....	225
➤ Fill in the blanks .....	226



## Important Questions

### UNIT - I

1. Explain evolution of databases.

*Ans :*

Refer Unit-I, Q.No. 3

2. List out various applications of database.

*Ans :*

Refer Unit-I, Q.No. 5

3. Write the differences between file management system and database management system.

*Ans :*

Refer Unit-I, Q.No. 10

4. Explain various techniques used to view data in database.

*Ans :*

Refer Unit-I, Q.No. 11

5. List and explain various types of database languages.

*Ans :*

Refer Unit-I, Q.No. 12

6. What are the differences between DBMS and RDBMS.

*Ans :*

Refer Unit-I, Q.No. 15

7. Why database design is important?

*Ans :*

Refer Unit-I, Q.No. 18

8. Explain the process of database designing.

*Ans :*

Refer Unit-I, Q.No. 19

9. Explain the major functional components of database.

*Ans :*

Refer Unit-I, Q.No. 20

10. Explain ACID Properties and how they are useful to transactions?

*Ans :*

Refer Unit-I, Q.No. 22

**11. Who is database administrator? What are the responsibilities of database administrator?**

*Ans :*

Refer Unit-I, Q.No. 27

**12. List out various types of database administrators?**

*Ans :*

Refer Unit-I, Q.No. 28

**13. What is mean by database schema?**

*Ans :*

Refer Unit-I, Q.No. 31

**14. What is Key ? how it is useful in database?**

*Ans :*

Refer Unit-I, Q.No. 38

**15. List out various types of relational query languages?**

*Ans :*

Refer Unit-I, Q.No. 42

**16. What is relational algebra? List and explain various types of relational operations.**

*Ans :*

Refer Unit-I, Q.No. 43

## UNIT - II

**1. Explain database designing process ?**

*Ans :*

Refer Unit-II, Q.No. 2

**2. What is relational integrity constraints in DBMS ?**

*Ans :*

Refer Unit-II, Q.No. 8

**3. Explain the process of creating ER Diagram into table with an example.**

*Ans :*

Refer Unit-II, Q.No. 18

**4. What are data modelling notations ?**

*Ans :*

Refer Unit-II, Q.No. 25

5. What are the other aspects of database design ?

*Ans :*

Refer Unit-II, Q.No. 26

---

6. What are the major objectives of good database design.

*Ans :*

Refer Unit-II, Q.No. 28

---

7. Explain first normal form (1NF) in detail.

*Ans :*

Refer Unit-II, Q.No. 34

---

8. Explain in detail Third Normal Form (3NF).

*Ans :*

Refer Unit-II, Q.No. 39

---

9. Explain in detail Boycecodd Normal Form (BCNF).

*Ans :*

Refer Unit-II, Q.No. 40

---

### UNIT - III

1. What is an operator in SQL ? Explain the various types of an SQL ?

*Ans :*

Refer Unit-III, Q.No. 5

---

2. List out various types of SQL Commands.

*Ans :*

Refer Unit-III, Q.No. 6

---

3. Explain in detail DCL Commands

*Ans :*

Refer Unit-III, Q.No. 9

---

4. Explain the use of as clause in SQL.

*Ans :*

Refer Unit-III, Q.No. 15

---

5. List and explain various set operations of SQL.

*Ans :*

Refer Unit-III, Q.No. 19

---

6. SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.

*Ans :*

Refer Unit-III, Q.No. 20

7. What is Join? List out various types of joins supported by SQL?

*Ans :*

Refer Unit-III, Q.No. 23

8. Explain inner join concept with an example.

*Ans :*

Refer Unit-III, Q.No. 24

9. What is Left Join explain with an example.

*Ans :*

Refer Unit-III, Q.No. 25

10. What is mean by view in SQL? Explain the process of creating, updating and dropping a view ?

*Ans :*

Refer Unit-III, Q.No. 29

11. What is transaction? Explain the properties of transaction ?

*Ans :*

Refer Unit-III, Q.No. 30

12. What is mean by constraints? List out most commonly used constraints in SQL?

*Ans :*

Refer Unit-III, Q.No. 32

13. Discuss in detail about SQL integrity constraints

*Ans :*

Refer Unit-III, Q.No. 33

14. List out various types of datatypes supported by SQL.

*Ans :*

Refer Unit-III, Q.No. 34

15. How can we access SQL from a prog-ramming language ?

*Ans :*

Refer Unit-III, Q.No. 36

**UNIT - IV**

1. State the architecture of transaction management.

*Ans :*

Refer Unit-IV, Q.No. 3

2. What is the purpose of concurrency control for a database?

*Ans :*

Refer Unit-IV, Q.No. 4

3. Define Serializability. State the importance of Serializability?

*Ans :*

Refer Unit-IV, Q.No. 6

4. Discuss in detail about the Recoverability of Schedules ?

*Ans :*

Refer Unit-IV, Q.No. 8

5. What is Lock? Explain various type of Locks in DBMS.

*Ans :*

Refer Unit-IV, Q.No. 9

6. Define deadlock. State the prevention and detection of deadlock?

*Ans :*

Refer Unit-IV, Q.No. 10

7. Discuss in detail about multi version timestamp ordering.

*Ans :*

Refer Unit-IV, Q.No. 12

8. Explain different Recovery Techniques/Approaches in DBMS ?

*Ans :*

Refer Unit-IV, Q.No. 18

9. Explain different Database Security Threats and How to Mitigate them?

*Ans :*

Refer Unit-IV, Q.No. 21

**10. Explain various Computer Based Controls for a multi user environment.**

*Ans :*

Refer Unit-IV, Q.No. 22

---

**11. What are the types of Access Control Mechanisms ?**

*Ans :*

Refer Unit-IV, Q.No. 25

---

**12. Explain briefly about Backup and Recovery and Integrity**

*Ans :*

Refer Unit-IV, Q.No. 28

---

**13. Explain the concept of Database Encryption and Decryption.**

*Ans :*

Refer Unit-IV, Q.No. 29

---

**14. Explain the concept of RAID.**

*Ans :*

Refer Unit-IV, Q.No. 30

# UNIT I

**Introduction:** Database-System Applications, Purpose of Database Systems, View of Data, Database Languages, Relational Databases, Database Design, Data Storage and Querying, Transaction Management, Database Architecture, Database Users and Administrators.

**Introduction to the Relational Model:** Structure of Relational Databases, Database Schema, Keys, Schema Diagrams, Relational Query Languages, Relational Operations.

## 1.1 INTRODUCTION TO DATABASE

### Q1. What is Data?

*Ans :*

Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

Word 'Data' is originated from the word 'datum' that means 'single piece of information.' It is plural of the word datum.

In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable.

### Q2. What is Database?

*Ans :*

A database is an organized collection of data, so that it can be easily accessed and managed.

Database handlers create a database in such a way that only one set of software program provides access of data to all the users.

The main purpose of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many dynamic websites on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.

There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.



### Q3. Explain evolution of databases.

*Ans :*

(Imp.)

The database has completed more than 50 years of journey of its evolution from flat-file system to relational and objects relational systems. It has gone through several generations.

#### Evolution

##### 1. File-Based

1968 was the year when File-Based database were introduced. In file-based databases, data was maintained in a flat file. Though files have many advantages, there are several limitations.

One of the major advantages is that the file system has various access methods, e.g., sequential, indexed, and random.

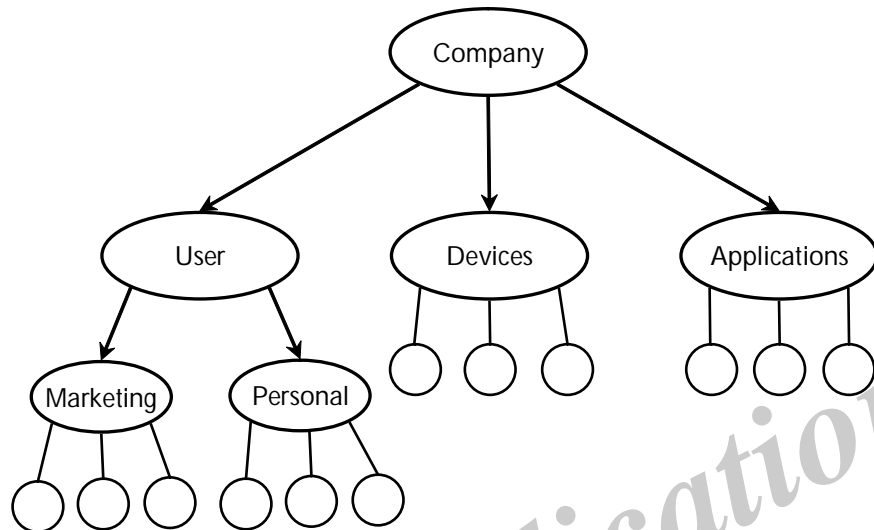
It requires extensive programming in a third-generation language such as COBOL, BASIC.

## 2. Hierarchical Data Model

1968-1980 was the era of the Hierarchical Database. Prominent hierarchical database model was IBM's first DBMS. It was called IMS (Information Management System).

In this model, files are related in a parent/child manner.

Below diagram represents Hierarchical Data Model. Small circle represents objects.



Like file system, this model also had some limitations like complex implementation, lack structural independence, can't easily handle a many-many relationship, etc.

## 3. Network data model

Charles Bachman developed the first DBMS at Honeywell called Integrated Data Store (IDS). It was developed in the early 1960s, but it was standardized in 1971 by the CODASYL group (Conference on Data Systems Languages).

In this model, files are related as owners and members, like to the common network model.

## 4. Network data model identified the following components

- Network schema (Database organization)
- Sub-schema (views of database per user)
- Data management language (procedural)

This model also had some limitations like system complexity and difficult to design and maintain.

## 5. Relational Database

### 1970 - Present

It is the era of Relational Database and Database Management. In 1970, the relational model was proposed by E.F. Codd.

Relational database model has two main terminologies called instance and schema.

The instance is a table with rows or columns

Schema specifies the structure like name of the relation, type of each column and name.

This model uses some mathematical concept like set theory and predicate logic.



The first internet database application had been created in 1995.

During the era of the relational database, many more models had introduced like object-oriented model, object-relational model, etc.

## 6. Cloud database

Cloud database facilitates you to store, manage, and retrieve their structured, unstructured data via a cloud platform. This data is accessible over the Internet. Cloud databases are also called a database as service (DBaaS) because they are offered as a managed service.

### Some best cloud options are

- AWS (Amazon Web Services)
- Snowflake Computing
- Oracle Database Cloud Services
- Microsoft SQL server
- Google cloud spanner

### Advantages

- i) **Lower costs:** Generally, company provider does not have to invest in databases. It can maintain and support one or more data centers.
- ii) **Automated:** Cloud databases are enriched with a variety of automated processes such as recovery, failover, and auto-scaling.
- iii) **Increased accessibility:** You can access your cloud-based database from any location, anytime. All you need is just an internet connection.

## 7. NoSQL Database

A NoSQL database is an approach to design such databases that can accommodate a wide variety of data models. NoSQL stands for "not only SQL." It is an alternative to traditional relational databases in which data is placed in tables, and data schema is perfectly designed before the database is built.

NoSQL databases are useful for a large set of distributed data.

Some examples of NoSQL database system with their category are:

- MongoDB, CouchDB, Cloudant (**Document-based**).
- Memcached, Redis, Coherence (**key-value store**).
- HBase, Big Table, Accumulo (**Tabular**).

### Advantage

#### i) High Scalability

NoSQL can handle an extensive amount of data because of scalability. If the data grows, NoSQL database scale it to handle that data in an efficient manner.

#### ii) High Availability

NoSQL supports auto replication. Auto replication makes it highly available because, in case of any failure, data replicates itself to the previous consistent state.

### Disadvantage

- i) **Open source:** NoSQL is an open-source database, so there is no reliable standard for NoSQL yet.
- ii) **Management challenge:** Data management in NoSQL is much more complicated than relational databases. It is very challenging to install and even more hectic to manage daily.
- iii) **GUI is not available:** GUI tools for NoSQL database are not easily available in the market.
- iv) **Backup:** Backup is a great weak point for NoSQL databases. Some databases, like MongoDB, have no powerful approaches for data backup.

## 8. The Object-Oriented Databases

The object-oriented databases contain data in the form of object and classes. Objects are the real-world entity, and types are the collection of objects. An object-oriented database is a combination of relational model features with objects oriented principles. It is an alternative implementation to that of the relational model.

Object-oriented databases hold the rules of object-oriented programming. An object-oriented database management system is a hybrid application.

The object-oriented database model contains the following properties.

**Object-oriented programming properties**

- Objects
- Classes
- Inheritance
- Polymorphism
- Encapsulation

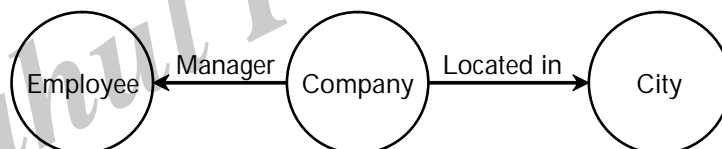
**Relational database properties**

- Atomicity
- Consistency
- Integrity
- Durability
- Concurrency
- Query processing

**9. Graph Databases**

A graph database is a NoSQL database. It is a graphical representation of data. It contains nodes and edges. A node represents an entity, and each edge represents a relationship between two edges. Every node in a graph database represents a unique identifier.

Graph databases are beneficial for searching the relationship between data because they highlight the relationship between relevant data.



Graph databases are very useful when the database contains a complex relationship and dynamic schema.

It is mostly used in supply chain management, identifying the source of IP telephony.

---

**Q4. Define DBMS. List out its advantages and disadvantages.**

*Ans :*

Database management System is software which is used to store and retrieve the database. For example, Oracle, MySQL, etc.; these are some popular DBMS tools.

- DBMS provides the interface to perform the various operations like creation, deletion, modification, etc.
- DBMS allows the user to create their databases as per their requirement.
- DBMS accepts the request from the application and provides specific data through the operating system.
- DBMS contains the group of programs which acts according to the user instruction.
- It provides security to the database.

**Advantage****i) Controls redundancy**

It stores all the data in a single database file, so it can control data redundancy.

**ii) Data sharing**

An authorized user can share the data among multiple users.

**iii) Backup**

It provides Backup and recovery subsystem. This recovery system creates automatic data from system failure and restores data if required.

**iv) Multiple user interfaces**

It provides a different type of user interfaces like GUI, application interfaces.

**Disadvantage****i) Size**

It occupies large disk space and large memory to run efficiently.

**ii) Cost**

DBMS requires a high-speed data processor and larger memory to run DBMS software, so it is costly.

**iii) Complexity**

DBMS creates additional complexity and requirements.

## 1.2 SYSTEM APPLICATIONS

**Q5. List out various applications of database.**

*Ans :*

**(Imp.)**

Applications where we use Database Management Systems are:

**i) Telecom**

There is a database to keep track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.

**ii) Industry**

Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.

**iii) Banking System**

For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.

**iv) Sales**

To store customer information, production information and invoice details.

**v) Airlines**

To travel through airlines, we make early reservations, this reservation information along with flight schedule is stored in database.

**vi) Education sector**

Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.

**vii) Online shopping**

You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

## 1.3 PURPOSE OF DATABASE SYSTEMS

**Q6. Explain the database concept of system.**

*Ans :*

Database systems arose in response to early methods of computerized management of commercial data. As an example of such methods,

typical of the 1960s, consider part of a university organization that, among other data, keeps information about all instructors, students, departments, and course offerings. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including programs to:

1. Add new students, instructors, and courses
2. Register students for courses and generate class rosters
3. Assign grades to students, compute grade point averages (GPA), and generate transcripts

System programmers wrote these application programs to meet the needs of the university.

New application programs are added to the system as the need arises. For example, suppose that a university decides to create a new major (say, computer science). As a result, the university creates a new department and creates new permanent files (or adds information to existing files) to record information about all the instructors in the department, students in that major, course offerings, degree requirements, etc. The university may have to write new application programs to deal with rules specific to the new major. New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs.

This typical file-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems. Keeping organizational information in a file-processing system has a number of major disadvantages:

### 1. Data redundancy and inconsistency

Since different programmers create the files and application programs over a long period, the various files are likely to have different structures

and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree. For example, a changed student address may be reflected in the Music department records but not elsewhere in the system.

### Difficulty in accessing data

Suppose that one of the university clerks needs to find out the names of all students who live within a particular postal-code area. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of all students. The university clerk has now two choices: either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same clerk needs to trim that list to include only those students who have taken at least 60 credit hours. As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory.

The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

### 2. Data isolation

Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

### 3. Integrity problems

The data values stored in the database must satisfy certain types of consistency constraints. Suppose the university maintains an account for each department, and records the balance amount in each account.

Suppose also that the university requires that the account balance of a department may never fall below zero. Developers enforce these constraints in the system by adding appropriate code in the various application programs.

However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

### 4. Atomicity problems

A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. Consider a program to transfer \$500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program, it is possible that the \$500 was removed from the balance of department A but was not credited to the balance of department B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur.

That is, the funds transfer must be atomic - it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

### 5. Concurrent-access anomalies

For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. Indeed, today, the largest Internet retailers may have millions of accesses per day to their data by shoppers. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. Consider department A, with an account balance of \$10,000. If two department clerks debit the account balance (by say \$500 and \$100, respectively) of department A at almost exactly the same time, the result of the concurrent executions may leave the budget in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value \$10,000, and write back \$9500 and \$9900, respectively. Depending on which one writes the value last, the account balance of department A may contain either \$9500 or \$9900, rather than the correct value of \$9400. To guard against this possibility, the system must maintain some form of supervision.

But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously. As another example, suppose a registration program maintains a count of students registered for a course, in order to enforce limits on the number of students registered. When a student registers, the program reads the current count for the courses, verifies that the count is not already at the limit, adds one to the count, and stores the count back in the database. Suppose two students register concurrently, with the count at (say) 39. The two program executions may both read the value 39, and both would then write back 40, leading to an incorrect increase of only 1, even though two students successfully registered for the course and the count should be 41. Furthermore, suppose the course registration limit was 40; in the above case both students would be able to register, leading to a violation of the limit of 40 students.

### 6. Security problems

Not every user of the database system should be able to access all the data. For example, in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.

**Q7. What is the purpose of DBMS ?***Ans :*

The Database Management System (DBMS) is defined as a software system that allows the user to define, create and maintain the database and provide control access to the data.

It is a collection of programs used for managing data and simultaneously it supports different types of users to create, manage, retrieve, update and store information.

**Purpose**

The purpose of DBMS is to transform the following:

- Data into information.
- Information into knowledge.
- Knowledge to the action.

The diagram given below explains the process as to how the transformation of data to information to knowledge to action happens respectively in the DBMS:

Previously, the database applications were built directly on top of the file system.

---

**Q8. State the drawbacks in file system.***Ans :*

There are so many drawbacks in using the file system. These are mentioned below:

- **Data redundancy and inconsistency:** Different file formats, duplication of information in different files.
- **Difficulty in accessing data:** To carry out new task we need to write a new program.
- **Data Isolation:** Different files and formats.
- Integrity problems.
- **Atomicity of updates:** Failures leave the database in an inconsistent state. For example, the fund transfer from one account to another may be incomplete.
- Concurrent access by multiple users.
- Security problems.

Database system offer so many solutions to all these problems

---

**Q9. State the uses of DBMS.***Ans :*

The main uses of DBMS are as follows:

- Data independence and efficient access of data.
- Application Development time reduces.
- Security and data integrity.
- Uniform data administration.
- Concurrent access and recovery from crashes.

**Q10. Write the differences between file management system and database management system.**

*Ans :*

**(Imp.)**

There are following differences between DBMS and File system:

DBMS	File System
DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	File system is a collection of data. In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details.	File system provides the detail of the data representation and storage of data.
DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure.	File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost.
DBMS provides a good protection mechanism.	It is very difficult to protect a file under the file system.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.
DBMS takes care of Concurrent access of data using some form of locking.	In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information.

#### 1.4 VIEW OF DATA

**Q11. Explain various techniques used to view data in database.**

*Ans :*

**(Imp.)**

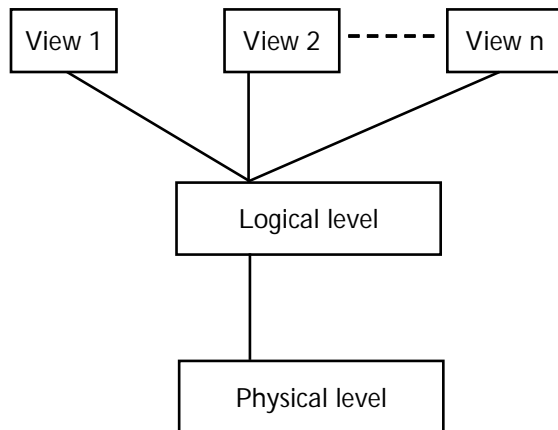
A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data. A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.

To fully understand the view of data, you must have a basic knowledge of data abstraction and instance & schema.

1. Data abstraction
2. Instance and schema

#### 1. Data abstraction

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.



Three Levels of data abstraction

We have three levels of abstraction:

### i) Physical level

This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.

### ii) Logical level

This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

### iii) View level

Highest level of data abstraction. This level describes the user interaction with database system.

### Example

Let's say we are storing customer information in a customer table. At physical level these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.

At the logical level these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

## 2. Instance and schema

### Schema

Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

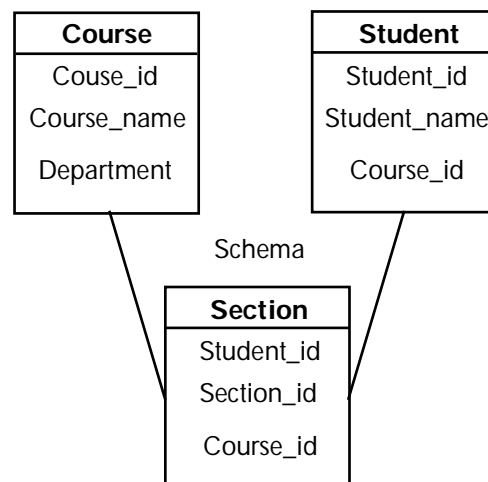
### For example

In the following diagram, we have a schema that shows the relationship between three tables: Course, Student and Section. The diagram only shows the design of the database, it doesn't show the data present in those tables. Schema is only a structural view (design) of a database as shown in the diagram below.

The design of a database at physical level is called physical schema, how the data stored in blocks of storage is described at this level.

Design of database at logical level is called logical schema, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).

Design of database at view level is called view schema. This generally describes end user interaction with database systems.



### Instance

The data stored in database at a particular moment of time is called instance of database.



Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

For example, let's say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Let's say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

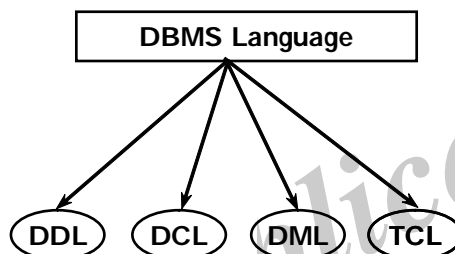
### 1.5 DATABASE LANGUAGES

**Q12. List and explain various types of database languages.**

*Ans :*

(Imp.)

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.



#### 1. Data Definition Language

- DDL stands for Data Definition Language. It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

#### 2. Data Manipulation Language

DML stands for Data Manipulation Language. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

### 3. Data Control Language

- DCL stands for data control language. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.  
(But in Oracle database, the execution of data control language does not have the feature of rolling back.)  
Here are some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

### 4. Transaction Control Language

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

## 1.6 RELATIONAL DATABASES

### Q13. Define relational database?

*Ans :*

(Imp.)

A relational database (RDB) is a collective set of multiple data sets organized by tables, records and columns. RDBs establish a well-defined relationship between database tables. Tables communicate and share information, which facilitates data searchability, organization and reporting.

RDBs use Structured Query Language (SQL), which is a standard user application that provides an easy programming interface for database interaction.

RDB is derived from the mathematical function concept of mapping data sets and was developed by Edgar F. Codd.

RDBs organize data in different ways. Each table is known as a relation, which contains one or more data category columns. Each table record (or row) contains a unique data instance defined for a corresponding column category. One or more data or record characteristics relate to one or many records to form functional dependencies. These are classified as follows:

- **One to One:** One table record relates to another record in another table.
- **One to Many:** One table record relates to many records in another table.
- **Many to One:** More than one table record relates to another table record.
- **Many to Many:** More than one table record relates to more than one record in another table.

RDBs have many other advantages, including:

- Easy extendibility, as new data may be added without modifying existing records. This is also known as scalability.
- New technology performance, power and flexibility with multiple data requirement capabilities.

- Data security, which is critical when data sharing is based on privacy. For example, management may share certain data privileges and access and block employees from other data, such as confidential salary or benefit information.

**Q14. What is RDBMS ? How it works ?**

*Ans :*

RDBMS stands for Relational Database Management Systems;

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.

It is called Relational Data Base Management System (RDBMS) because it is based on relational model introduced by E.F. Codd.

Data is represented in terms of tuples (rows) in RDBMS. Relational database is most commonly used database. It contains number of tables and each table has its own primary key.

Due to a collection of organized set of tables, data can be accessed easily in RDBMS.

**Q15. What are the differences between DBMS and RDBMS.**

*Ans :*

(Imp.)

The main differences between DBMS and RDBMS are given below:

S.No.	DBMS	RDBMS
1.	DBMS applications store data as file.	RDBMS applications store data in a tabular form.
2.	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3.	Normalization is not present in DBMS.	Normalization is present in RDBMS.
4.	DBMS does not apply any security with regards to data manipulation.	RDBMS defines the integrity constraint for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
5.	DBMS uses file system to store data, so there will be no relation between the tables.	in RDBMS, data values are stored in the form of tables, so a relationship between these data values will be stored in the form of a table as well.
6.	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7.	DBMS does not support distributed database.	RDBMS supports distributed database.
8.	DBMS is meant to be for small organization and deal with small data. it supports single user.	RDBMS is designed to handle large amount of data. it supports multiple users.
9.	Examples of DBMS are file systems, xml etc.	Example of RDBMS are mysql, postgre, sql server, oracle etc.

**Q16. List and explain various types of databases.**

*Ans :*

Database technology has changed and evolved over the years. Relational, NoSQL, hierarchical...it can start to get confusing. Storing data doesn't have to be a headache. If you're trying to pick the right database for your organization, here's a guide to the properties and uses of each type.

**1. Relational databases**

Relational databases have been around since the 1970s. The name comes from the way that data is stored in multiple, related tables. Within the tables, data is stored in rows and columns. The relational database management system (RDBMS) is the program that allows you to create, update, and administer a relational database. Structured Query Language (SQL) is the most common language for reading, creating, updating and deleting data. Relational databases are very reliable. They are compliant with ACID (Atomicity, Consistency, Isolation, Durability), which is a standard set of properties for reliable database transactions. Relational databases work well with structured data. Organizations that have a lot of unstructured or semi-structured data should not be considering a relational database.

**Examples:** Microsoft SQL Server, Oracle Database, MySQL, PostgreSQL and IBM Db2

**2. NoSQL databases**

NoSQL is a broad category that includes any database that doesn't use SQL as its primary data access language. These types of databases are also sometimes referred to as non-relational databases. Unlike in relational databases, data in a NoSQL database doesn't have to conform to a pre-defined schema, so these types of databases are great for organizations seeking to store unstructured or semi-structured data. One advantage of

NoSQL databases is that developers can make changes to the database on the fly, without affecting applications that are using the database.

**Examples:** Apache Cassandra, MongoDB, CouchDB, and CouchBase

**3. Cloud databases**

A cloud database refers to any database that's designed to run in the cloud. Like other cloud-based applications, cloud databases offer flexibility and scalability, along with high availability. Cloud databases are also often low-maintenance, since many are offered via a SaaS model.

**Examples:** Microsoft Azure SQL Database, Amazon Relational Database Service, Oracle Autonomous Database.

**4. Columnar databases**

Also referred to as column data stores, columnar databases store data in columns rather than rows. These types of databases are often used in data warehouses because they're great at handling analytical queries. When you're querying a columnar database, it essentially ignores all of the data that doesn't apply to the query, because you can retrieve the information from only the columns you want.

**Examples:** Google BigQuery, Cassandra, HBase, MariaDB, Azure SQL Data Warehouse

**5. Wide column databases**

Wide column databases, also known as wide column stores, are schema-agnostic. Data is stored in column families, rather than in rows and columns. Highly scalable, wide column databases can handle petabytes of data, making them ideal for supporting real-time big data applications.

**Examples:** BigTable, Apache Cassandra and Scylla

**6. Object-oriented databases**

An object-oriented database is based on object-oriented programming, so data and

all of its attributes, are tied together as an object. Object-oriented databases are managed by object-oriented database management systems (OODBMS). These databases work well with object-oriented programming languages, such as C++ and Java. Like relational databases, object-oriented databases conform to ACID standards.

**Examples:** Wakanda, ObjectStore

## 7. Key-value databases

One of the simplest types of NoSQL databases, key-value databases save data as a group of key-value pairs made up of two data items each. They're also sometimes referred to as a key-value store. Key-value databases are highly scalable and can handle high volumes of traffic, making them ideal for processes such as session management for web applications, user sessions for massive multi-player online games, and online shopping carts.

**Examples:** Amazon DynamoDB, Redis

## 8. Hierarchical databases

Hierarchical databases use a parent-child model to store data. If you were to draw a picture of a hierarchical database, it would look like a family tree, with one object on top branching down to multiple objects beneath it. The one-to-many format is rigid, so child records can't have more than one parent record. Originally developed by IBM in the early 1960s, hierarchical databases are commonly used to support high-performance and high availability applications.

**Examples:** IBM Information Management System (IMS), Windows Registry

## 9. Document databases

Document databases, also known as document stores, use JSON-like documents to model data instead of rows and columns. Sometimes referred to as document-oriented databases, document databases are designed to store and manage document-oriented information, also referred to as semi-

structured data. Document databases are simple and scalable, making them useful for mobile apps that need fast iterations.

**Examples:** MongoDB, Amazon Document DB, Apache CouchDB

## 10. Graph databases

Graph databases are a type of NoSQL database that are based on graph theory. Graph-Oriented Database Management Systems (DBMS) software is designed to identify and work with the connections between data points. Therefore graph databases are often used to analyze the relationships between heterogeneous data points, such as in fraud prevention or for mining data about customers from social media.

**Examples:** Datastax Enterprise Graph, Neo4J

## 11. Time series databases

A time series database is a database optimized for time-stamped, or time series, data. Examples of this type of data include network data, sensor data, and application performance monitoring data. All of those Internet of Things sensors that are getting attached to everything put out a constant stream of time series data.

**Examples:** Druid, eXtremeDB, InfluxDB

# 1.7 DATABASE DESIGN

## Q17. What is database design?

*Ans :*

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

### Q18. Why database design is important?

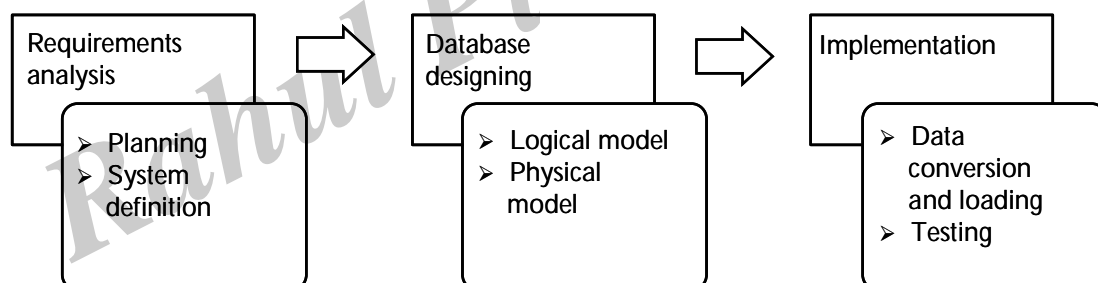
*Ans :*

(Imp.)

The important consideration that can be taken into account while emphasizing the importance of database design can be explained in terms of the following points given below.

1. Database designs provide the blueprints of how the data is going to be stored in a system. A proper design of a database highly affects the overall performance of any application.
2. The designing principles defined for a database give a clear idea of the behavior of any application and how the requests are processed.
3. Another instance to emphasize the database design is that a proper database design meets all the requirements of users.
4. Lastly, the processing time of an application is greatly reduced if the constraints of designing a highly efficient database are properly implemented.

### Life Cycle



Although, the life cycle of a database is not an important discussion that has to be taken forward in this article because we are focused on the database design. But, before jumping directly on the designing models constituting database design it is important to understand the overall workflow and life-cycle of the database.

### 1. Requirement Analysis

First of all, the planning has to be done on what are the basic requirements of the project under which the design of the database has to be taken forward. Thus, they can be defined as:-

- i) Planning:** This stage is concerned with planning the entire DDLC (Database Development Life Cycle). The strategic considerations are taken into account before proceeding.
- ii) System definition:** This stage covers the boundaries and scopes of the proper database after planning.

## 2. Database Designing

The next step involves designing the database considering the user-based requirements and splitting them out into various models so that load or heavy dependencies on a single aspect are not imposed. Therefore, there has been some model-centric approach and that's where logical and physical models play a crucial role.

- i) **Physical Model:** The physical model is concerned with the practices and implementations of the logical model.
- ii) **Logical Model:** This stage is primarily concerned with developing a model based on the proposed requirements. The entire model is designed on paper without any implementation or adopting DBMS considerations.

## 3. Implementation

The last step covers the implementation methods and checking out the behavior that matches our requirements. It is ensured with continuous integration testing of the database with different data sets and conversion of data into machine understandable language. The manipulation of data is primarily focused on these steps where queries are made to run and check if the application is designed satisfactorily or not.

- i) **Data conversion and loading:** This section is used to import and convert data from the old to the new system.
- ii) **Testing:** This stage is concerned with error identification in the newly implemented system. Testing is a crucial step because it checks the database directly and compares the requirement specifications.

---

### Q19. Explain the process of database designing.

*Ans :*

(Imp.)

#### Database Design Process

The process of designing a database carries various conceptual approaches that are needed to be kept in mind. An ideal and well-structured database design must be able to:

1. Save disk space by eliminating redundant data.
2. Maintains data integrity and accuracy.
3. Provides data access in useful ways.
4. Comparing Logical and Physical data models.

#### Logical

A logical data model generally describes the data in as many details as possible, without having to be concerned about the physical implementations in the database. Features of logical data model might include:

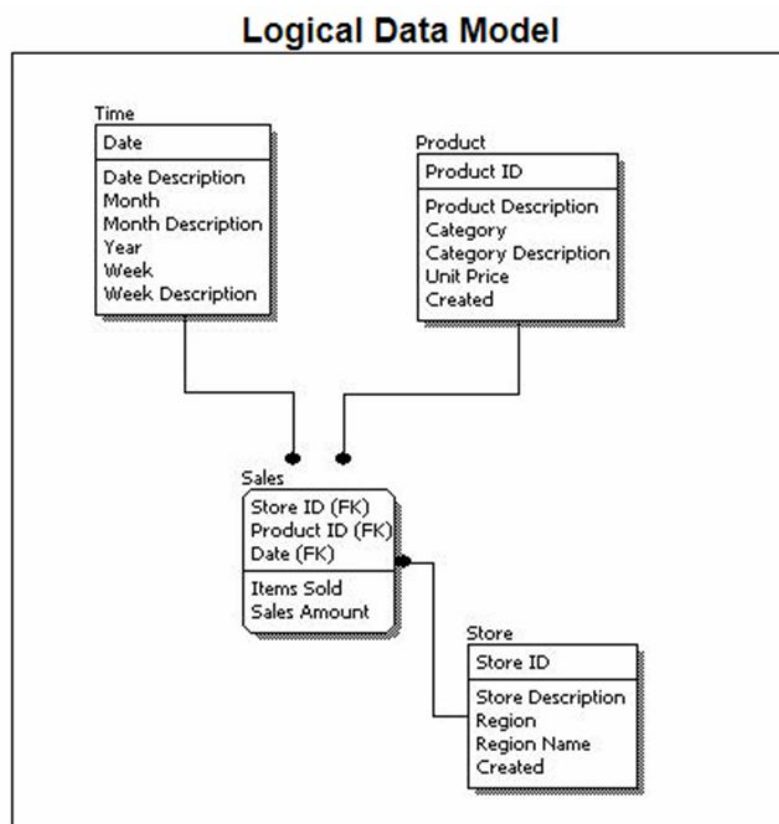
1. All the entities and relationships amongst them.
2. Each entity has well-specified attributes.
3. The primary key for each entity is specified.
4. Foreign keys which are used to identify a relationship between different entities are specified.
5. Normalization occurs at this level.

A logical model can be designed using the following approach:

1. Specify all the entities with primary keys.
2. Specify concurrent relationships between different entities.
3. Figure out each entity attributes
4. Resolve many-to-many relationships.
5. Carry out the process of normalization.

Also, one important factor after following the above approach is to critically examine the design based on requirement gathering. If the above steps are strictly followed, there are chances of creating a highly efficient database design that follows the native approach.

To understand these points, see the image below to get a clear picture.



If we compare the logical data model as shown in the figure above with some sample data in the diagram, we can come up with facts that in a conceptual data model there are no presence of a primary key whereas a logical data model has primary keys for all of its attributes. Also, logical data model the cover relationship between different entities and carries room for foreign keys to establish relationships among them.

### Physical

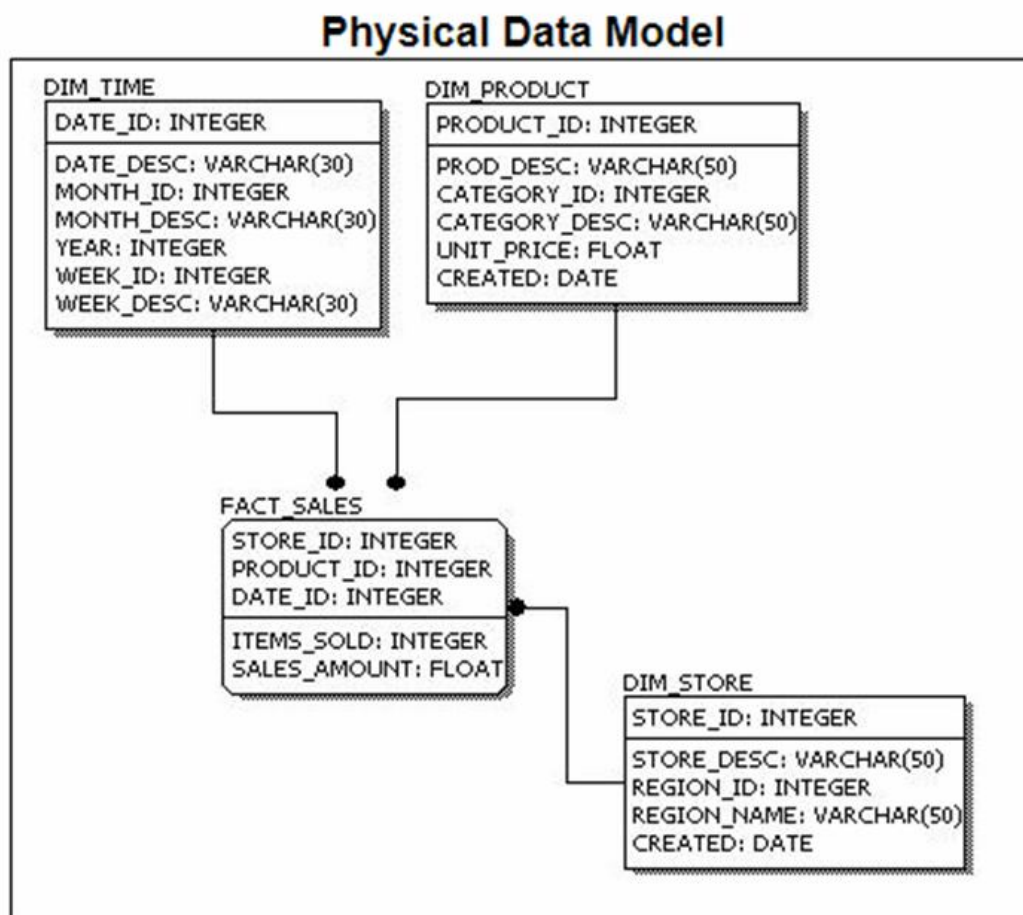
A Physical data mode generally represents how the approach or concept of designing the database. The main purpose of the physical data model is to show all the structures of the table including the column name, column data type, constraints, keys(primary and foreign), and the relationship among tables. The following are the features of a physical data model:



1. Specifies all the columns and tables.
2. Specifies foreign keys that usually define the relationship between tables.
3. Based on user requirements, de-normalization might occur.
4. Since the physical consideration is taken into account so there will straightforward reasons for difference than a logical model.
5. Physical models might be different for different RDBMS. For example, the data type column may be different in MySQL and SQL Server.

While designing a physical data model, the following points should be taken into consideration:

1. Convert the entities into tables.
2. Convert the defined relationships into foreign keys.
3. Convert the data attributes into columns.
4. Modify the data model constraints based on physical requirements.



Comparing this physical data model with the logical with the previous logical model, we might conclude the differences that in a physical database entity names are considered table names and attributes are considered column names. Also, the data type of each column is defined in the physical model depending on the actual database used.

**1.8 DATA STORAGE AND QUERYING**

**Q20. Explain the major functional components of database.**

*Ans :*

**(Imp.)**

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data. A gigabyte is approximately 1000 megabytes (actually 1024) (1 billion bytes), and a terabyte is 1 million megabytes (1 trillion bytes). Since the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed. Since the movement of data to and from disk is slow relative to the speed of the central processing unit, it is imperative that the database system structure the data so as to minimize the need to move data between disk and main memory.

The query processor is important because it helps the database system to simplify and facilitate access to data. The query processor allows database users to obtain good performance while being able to work at the view level and not be burdened with understanding the physical-level details of the implementation of the system. It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.

### 1. Storage Manager

The storage manager is the component of a database system that provides the 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 the interaction with the file manager. The raw data are stored on the disk using the file system

provided by the operating system. The storage manager translates the various DML statements into low-level file-system commands.

Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

- **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
- **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

The storage manager implements several data structures as part of the physical system implementation:

- **Data files**, which store the database itself.
- **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
- **Indices**, which can provide fast access to data items. Like the index in this textbook, a database index provides pointers to those data items that hold a particular value. For example, we could use an index to find the instructor record with a particular ID, or all instructor records with a particular name.

Hashing is an alternative to indexing that is faster in some but not all cases.

**2. The Query Processor**

The query processor components include:

- **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
  - **DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
- A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization; that is, it picks the lowest cost evaluation plan from among the alternatives.
- **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.

**1.9 TRANSACTION MANAGEMENT****Q21. What is Transaction?**

(OR)

**Define the term transaction.***Ans :*

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs. 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

**A's Account**

```
Open_Account(A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account(A)
```

**B's Account**

```
Open_Account(B)
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account(B)
```

**Q22. Explain ACID Properties and how they are useful to transactions?***Ans :*

(Imp.)

A transaction is a very small unit of a program and it may contain several low level tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability " commonly known as ACID properties " in order to ensure accuracy, completeness, and data integrity.

- (i) **Atomicity:** This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
- (ii) **Consistency:** The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
- (iii) **Durability:** The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
- (iv) **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.

**Q23. List out various states of a transaction**

*Ans :*

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

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

**Q24. Explain in detail about transaction management.**

*Ans :*

A **transaction** is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database consistency constraints. That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates.

However, during the execution of a transaction, it may be necessary temporarily to allow

inconsistency, since either the debit of A or the credit of B must be done before the other. This temporary inconsistency, although necessary, may lead to difficulty if a failure occurs.

It is the programmer's responsibility to define properly the various transactions, so that each preserves the consistency of the database. For example, the transaction to transfer funds from the account of department A to the account of department B could be defined to be composed of two separate programs: one that debits account A, and another that credits account B. The execution of these two programs one after the other will indeed preserve consistency. However, each program by itself does not transform the database from a consistent state to a new consistent state. Thus, those programs are not transactions.

Ensuring the atomicity and durability properties is the responsibility of the database system itself - specifically, of the recovery manager. In the absence of failures, all transactions complete successfully, and atomicity is achieved easily.

However, because of various types of failure, a transaction may not always complete its execution successfully. If we are to ensure the atomicity property, a failed transaction must have no effect on the state of the database. Thus, the database must be restored to the state in which it was before the transaction in question started executing. The database system must therefore perform failure recovery, that is, detect system failures and restore the database to the state that existed prior to the occurrence of the failure.

Finally, when several transactions update the database concurrently, the consistency of data may no longer be preserved, even though each individual transaction is correct. It is the responsibility of the concurrency-control manager to control the interaction among the concurrent transactions, to ensure the consistency of the database. The transaction manager consists of the concurrency-control manager and the recovery manager.

The concept of a transaction has been applied broadly in database systems and applications. While the initial use of transactions was in financial applications, the concept is now used in real-time applications in telecommunication, as well as in the management of long-duration activities such as product designer administrative workflows.

### 1.10 DATABASE ARCHITECTURE

**Q25. What do you mean by Database Architecture? State various types of database architecture.**

*Ans :*

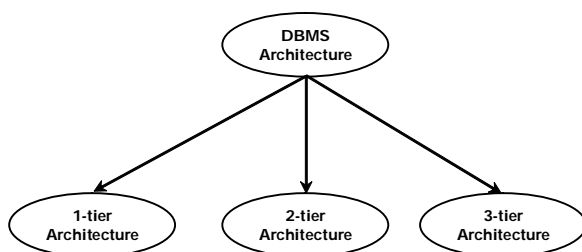
A Database Architecture is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered. It also helps to understand the components of a database.

A Database stores critical information and helps access data quickly and securely. Therefore, selecting the correct Architecture of DBMS helps in easy and efficient data management.

- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database to get their request done.

#### Types

Database architecture can be seen as a single tier or multitier. But logically, database architecture is of two types like: 2-tier architecture and 3-tier architecture.



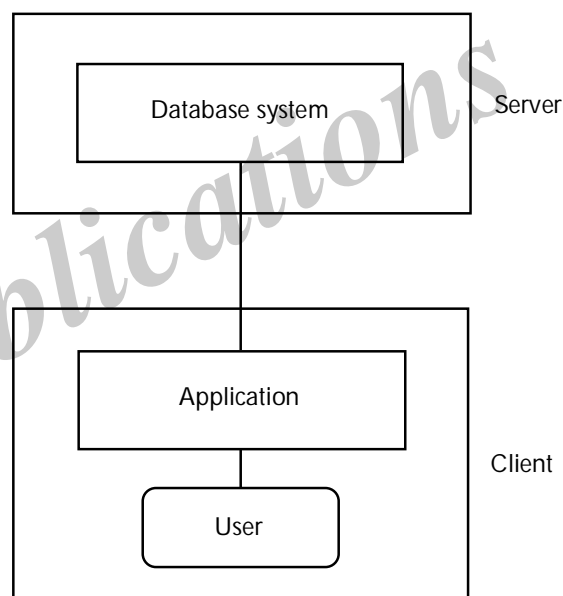
#### i) 1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.

- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

#### ii) 2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: ODBC, JDBC are used.



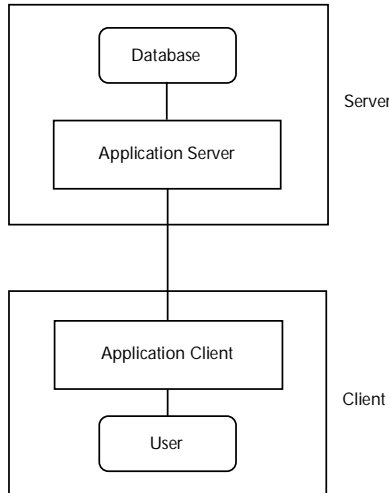
**Fig: 2-tier Architecture**

- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

#### iii) 3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.

- The application on the client-end interacts with an application server which further communicates with the database system.



**Fig: 3-tier Architecture**

- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

### 1.11 DATABASE USERS AND ADMINISTRATORS

**Q26. List out various types of database users.**

*Ans :*

Database users are the ones who really use and take the benefits of the database. There will be different types of users depending on their needs and way of accessing the database.

#### 1. Application Programmers

They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal, etc. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in a particular department will involve a query to fetch the data from the database. It will include an embedded SQL query in the C Program.

#### 2. Sophisticated Users

They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of a query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirements. In short, we can say this category includes designers and developers of DBMS and SQL.

#### 3. Specialized Users

These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement.

#### 4. Stand-alone Users

These users will have a stand-alone database for their personal use. These kinds of the database will have readymade database packages which will have menus and graphical interfaces.

#### 5. Native Users

These are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfill their requests.

**Q27. Who is database administrator? What are the responsibilities of database administrator?**

*Ans :*

**(Imp.)**

The life cycle of a database starts from designing, implementing to the administration of it. A database for any kind of requirement needs to be designed perfectly so that it should work without any issues. Once all the design is complete, it needs to be installed. Once this step is complete, users start using the database. The database grows as the data grows in the database. When the database becomes huge, its performance comes down. Also accessing the data from the database becomes a challenge. There will be unused memory in the database,

making the memory inevitably huge. This administration and maintenance of the database are taken care of by the database Administrator – DBA.

A DBA has many responsibilities. A good-performing database is in the hands of DBA.

- **Installing and upgrading the DBMS Servers:** DBA is responsible for installing a new DBMS server for the new projects. He is also responsible for upgrading these servers as there are new versions that come into the market or requirement. If there is any failure in the up-gradation of the existing servers, he should be able to revert the new changes back to the older version, thus maintaining the DBMS working. He is also responsible for updating the service packs/ hotfixes/ patches to the DBMS servers.
- **Design and implementation:** Designing the database and implementing is also DBA's responsibility. He should be able to decide on proper memory management, file organizations, error handling, log main-tenance, etc for the database.
- **Performance tuning:** Since the database is huge and it will have lots of tables, data, constraints, and indices, there will be variations in the performance from time to time. Also, because of some designing issues or data growth, the database will not work as expected. It is the responsibility of the DBA to tune the database performance. He is responsible to make sure all the queries and programs work in a fraction of seconds.
- **Migrate database servers:** Sometimes, users using oracle would like to shift to SQL server or Netezza. It is the responsibility of DBA to make sure that migration happens without any failure, and there is no data loss.
- **Backup and Recovery:** Proper backup and recovery programs needs to be developed by DBA and has to be maintained him. This is one of the main responsibilities of DBA. Data/objects should be backed up regularly so that if there is any crash, it should be recovered without much effort and data loss.
- **Security:** DBA is responsible for creating various database users and roles, and giving them different levels of access rights.
- **Documentation:** DBA should be properly documenting all his activities so that if he quits or any new DBA comes in, he should be able to understand the database without any effort. He should basically maintain all his installation, backup, recovery, security methods. He should keep various reports about database performance.

---

**Q28. List out various types of database administrators?**

*Ans .:*

**(Imp.)**

There are different kinds of DBA depending on the responsibility that he owns.

- **Administrative DBA :** This DBA is mainly concerned with installing, and maintaining DBMS servers. His prime tasks are installing, backups, recovery, security, replications, memory management, configurations, and tuning. He is mainly responsible for all administrative tasks of a database.
- **Development DBA:** He is responsible for creating queries and procedures for the requirement. Basically, his task is similar to any database developer.
- **Database Architect:** Database architect is responsible for creating and maintaining the users, roles, access rights, tables, views, constraints, and indexes. He is mainly responsible for designing the structure of the database depending on the requirement. These structures will be used by developers and development DBA to code.

- **Data Warehouse DBA:** DBA should be able to maintain the data and procedures from various sources in the data warehouse. These sources can be files, COBOL, or any other programs. Here data and programs will be from different sources. A good DBA should be able to keep the performance and function levels from these sources at the same pace to make the data warehouse work.
- **Application DBA:** He acts like a bridge between the application program and the database. He makes sure all the application program is optimized to interact with the database. He ensures all the activities from installing, upgrading, and patching, maintaining, backup, recovery to executing the records work without any issues.
- **OLAP DBA:** He is responsible for installing and maintaining the database in OLAP systems. He maintains only OLAP databases.

### 1.12 INTRODUCTION TO THE RELATIONAL MODEL

**Q29. What is relational model ?**

*Ans :*

Relational Model (RM) represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

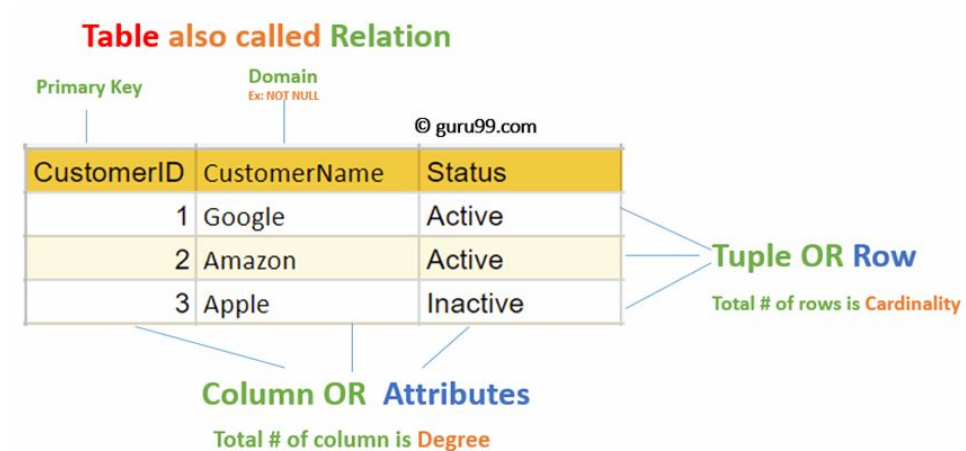
The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

**Q30. Discuss basic relational model concepts.**

*Ans :*

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student\_Rollno, NAME, etc.
2. **Tables:** In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple:** It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance:** Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key:** Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain:** Every attribute has some pre-defined value and scope which is known as attribute domain





### Q31. Explain in detail various types of integrity constraints?

*Ans :*

(Imp.)

Relational Integrity constraints in DBMS are referred to conditions which must be present for a valid relation. These Relational constraints in DBMS are derived from the rules in the mini-world that the database represents.

There are many types of Integrity Constraints in DBMS. Constraints on the Relational database management system is mostly divided into three main categories are:

1. Domain Constraints
2. Key Constraints
3. Referential Integrity Constraints

#### 1. Domain Constraints

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

##### Example

```
Create DOMAIN CustomerName
CHECK (value not NULL)
```

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

#### 2. Key Constraints

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.

**Example**

In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID = 1 is only for the CustomerName = "Google".

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

**3. Referential Integrity Constraints**

Referential Integrity constraints in DBMS are based on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

**Example**

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Customer

InvoiceNo	CustomerID	Amount
1	1	\$100
2	1	\$200
3	2	\$150

Billing

In the above example, we have 2 relations, Customer and Billing.

**Q32. List out various operations of relational model?**

*Ans ;*

**Operations in Relational Model**

Four basic update operations performed on relational database model are

Insert, update, delete and select.


- Insert is used to insert data into the relation
- Delete is used to delete tuples from the table.
- Update allows you to change the values of some attributes in existing tuples.
- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

### Insert Operation

The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

### Update Operation

You can see that in the below-given relation table CustomerName = 'Apple' is updated from Inactive to Active.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active




CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

### Delete Operation

To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

In the above-given example, CustomerName = "Apple" is deleted from the table.

The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from other tuples in the same database.

### Select Operation

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active



CustomerID	CustomerName	Status
2	Amazon	Active

In the above-given example, CustomerName = "Amazon" is selected

### Q33. What are the advantages and disadvantages of relational data model ?

*Ans :*

#### Advantages

##### i) Simplicity

A Relational data model in DBMS is simpler than the hierarchical and network model.

**ii) Structural Independence**

The relational database is only concerned with data and not with a structure. This can improve the performance of the model.

**iii) Easy to use**

The Relational model in DBMS is easy as tables consisting of rows and columns are quite natural and simple to understand

**iv) Query capability**

It makes possible for a high-level query language like SQL to avoid complex database navigation.

**v) Data independence**

The Structure of Relational database can be changed without having to change any application.

**vi) Scalable**

Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

**Disadvantages**

- Few relational databases have limits on field lengths which can't be exceeded.
- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.
- Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another.

**1.13 STRUCTURE OF RELATIONAL DATABASES**

**Q34. What is the basic structure of a relational database ?**

*Ans :*

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

Account-number	Branch-name	Balance
A-101	Downtown	500
A-102	Porryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

**Fig.: The account relation**

A row in a table represents a relationship among a set of values. Since a table is a collection of such relationships, there is a close correspondence between the concept of table and the mathematical concept of relation, from which the relational data model takes its name. In what follows, we introduce the concept of relation.

**Basic Structure**

Consider the account table of above Figure. It has three column headers: account-number, branch-name, and balance. Following the terminology of the relational model, these headers are attributes. For each attribute, there is a set of permitted values, called the domain of that attribute. For the attribute branch-name, for example, the domain is the set of all branch names.

Let  $D_1$  denote the set of all account numbers,  $D_2$  the set of all branch names, and  $D_3$  the set of all balances. Any row of account must consist of a 3-tuple  $(v_1, v_2, v_3)$ , where  $v_1$  is an account number (that is,  $v_1$  is in domain  $D_1$ ),  $v_2$  is a branch name (that is,  $v_2$  is in domain  $D_2$ ), and  $v_3$  is a balance (that is,  $v_3$  is in domain  $D_3$ ). In general, account will contain only a subset of the set of all possible rows.

Account-number	Branch-name	Balance
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750

**Fig.: The account relation with unordered tuples**

Therefore, account is a subset of,

$$D_1 \times D_2 \times D_3$$

In general, a **table** of  $n$  attributes must be a subset of

$$D_1 \times D_2 \times \dots \times D_{n-1} \times D_n$$

Mathematicians define a **relation** to be a subset of a Cartesian product of a list of domains. This definition corresponds almost exactly with our definition of table. The only difference is that we have assigned names to attributes, whereas mathematicians rely on numeric "names," using the integer 1 to denote the attribute whose domain appears first in the list of domains, 2 for the attribute whose domain appears second, and so on. Because tables are essentially relations, we shall use the mathematical terms relation and tuple in place of the terms table and row. A tuple variable is a variable that stands for a tuple; in other words, a tuple variable is a variable whose domain is the set of all tuples.

In the account relation of above Figure, there are seven tuples. Let the tuple variable  $t$  refer to the first tuple of the relation. We use the notation  $t[\text{account-number}]$  to denote the value of  $t$  on the account-number attribute. Thus,  $t[\text{account-number}] = \text{"A-101,"}$  and  $t[\text{branch-name}] = \text{"Downtown"}$ . Alternatively, we may write  $t[1]$  to denote the value of tuple  $t$  on the first attribute (account-number),  $t[2]$  to denote branch-name, and so on. Since a relation is a set of tuples, we use the mathematical notation of  $t \in r$  to denote that tuple  $t$  is in relation  $r$ .

The order in which tuples appear in a relation is irrelevant, since a relation is a set of tuples. Thus, whether the tuples of a relation are listed in sorted order, as in Figure above or are unsorted, as in Figure, does not matter; the relations in the two figures above are the same, since both contain the same set of tuples.

For all relations  $r$ , the domains of all attributes of  $r$  be atomic. A domain is atomic if elements of the domain are considered to be indivisible units. For example, the set of integers is an atomic domain, but the set of all sets of integers is a non atomic domain.

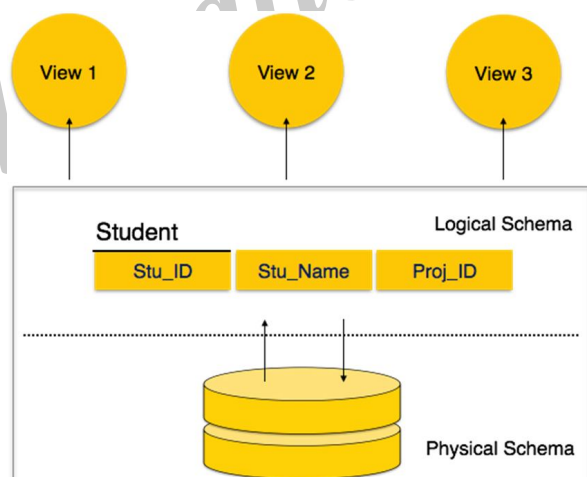
## 1.14 DATABASE SCHEMA

**Q35. What is mean by database schema?**

*Ans :* (Imp.)

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.



A database schema can be divided broadly into two categories:

### i) Physical Database Schema

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

### ii) Logical Database Schema

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

**Q36. What is Database instance ?***Ans :*

It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

**Q37. List out Schema integration requirements.***Ans :*

It can be useful to integrate multiple sources into a single schema. Make sure these requirements are met for a seamless transition:

**i) Overlap preservation**

Every overlapping element in the schemas you are integrating should be in a database schema table.

**ii) Extended overlap preservation**

Elements that only appear in one source, but that are associated with overlapping elements, should be copied to the resulting database schema.

**iii) Normalization**

Independent relationships and entities should not be lumped together in the same table in the database schema.

**iv) Minimality**

It's ideal if none of the elements in any of the sources are lost.

**1.15 KEYS****Q38. What is Key ? how it is useful in database?***Ans :***(Imp.)**

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

**For example**

In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport\_number, license\_number, SSN are keys since they are unique for each person.

STUDENT	PERSON
ID	Name
Name	DOB
Address	Passport_Number
Course	License_Number
	SSN

**Q39. List out various types of Keys.***Ans :***Primary Key**

A primary is a column or set of columns in a table that uniquely identifies tuples (rows) in that table.

**Super Key**

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

**Candidate Key**

A super key with no redundant attribute is known as candidate key

**Alternate Key**

Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate or secondary keys.

**Composite Key**

A key that consists of more than one attribute to uniquely identify rows (also known as records & tuples) in a table is called composite key.

**Foreign Key**

Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

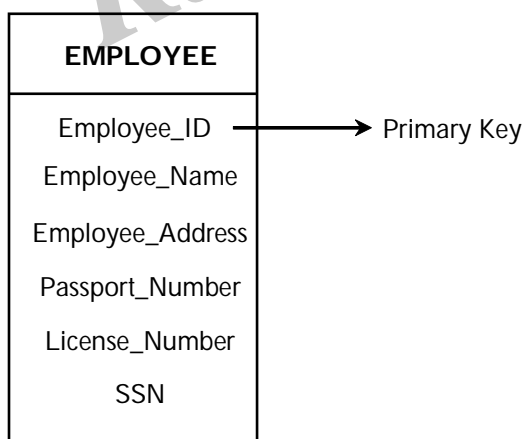
**Q40. Discuss the following keys in detail.**

- a) Primary key
- b) candidate key
- c) Super key
- d) foreign key

*Ans :*

**(a) Primary key**

- It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.



- In the EMPLOYEE table, ID can be primary key since it is unique for each

employee. In the EMPLOYEE table, we can even select License\_Number and Passport\_Number as primary key since they are also unique.

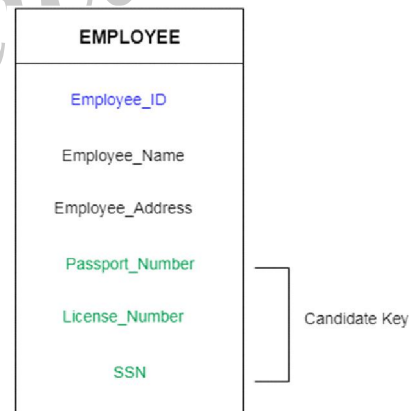
- For each entity, selection of the primary key is based on requirement and developers.

**b) Candidate key**

- A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key.

**For example**

In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport\_Number, and License\_Number, etc. are considered as a candidate key.

**c) Super Key**

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

**For example**

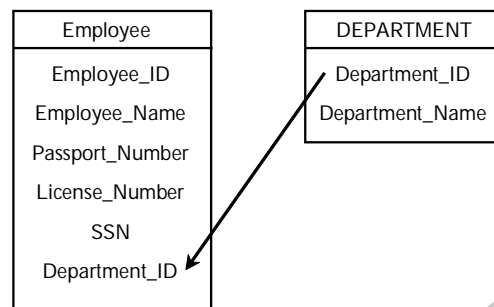
In the above EMPLOYEE table, for (EMPLOYEE\_ID, EMPLOYEE\_NAME) the name of two employees can be the same, but their EMPLOYEE\_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID, (EMPLOYEE\_ID, EMPLOYEE-NAME), etc.



**d) Foreign key**

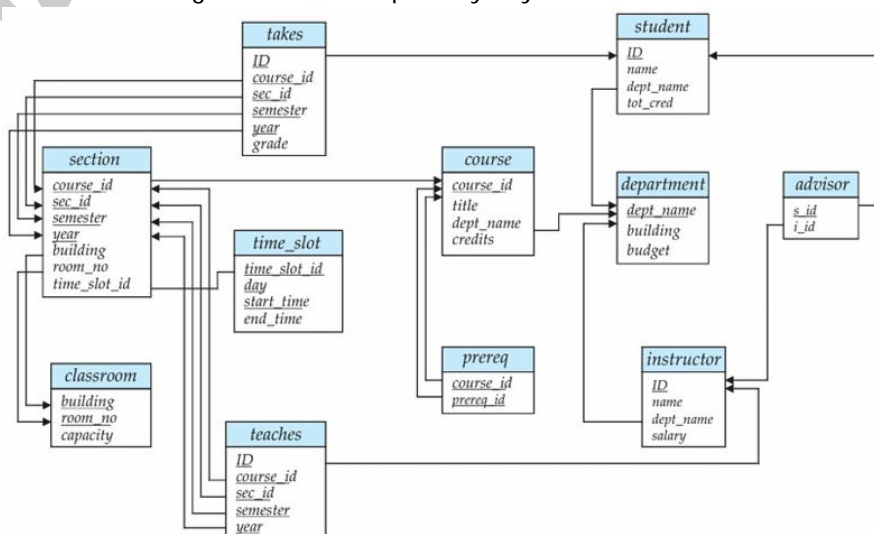
- Foreign keys are the column of the table which is used to point to the primary key of another table.
- In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department\_Id as a new attribute in the EMPLOYEE table.
- Now in the EMPLOYEE table, Department\_Id is the foreign key, and both the tables are related.

**1.16 SCHEMA DIAGRAMS**

**Q41. Draw and explain schema diagram for University Database.**

*Ans :*

A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams. Following Figure shows the schema diagram for our university organization. Each relation appears as a box, with the relation name at the top in blue, and the attributes listed inside the box. Primary key attributes are shown underlined. Foreign key dependencies appear as arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation.



**Figure: Schema diagram for the university database**



Referential integrity constraints other than foreign key constraints are not shown explicitly in schema diagrams. Entity relationship diagrams let us represent several kinds of constraints, including general referential integrity constraints. Many database systems provide design tools with a graphical user interface for creating schema diagrams.

### 1.17 RELATIONAL QUERY LANGUAGES AND RELATIONAL OPERATIONS

**Q42. List out various types of relational query languages?**

*Ans :*

(Imp.)

Relational query languages use relational algebra to break the user requests and instruct the DBMS to execute the requests. It is the language by which user communicates with the database. These relational query languages can be procedural or non-procedural.

- i) **Procedural Query Language:** A procedural query language will have set of queries instructing the DBMS to perform various transactions in the sequence to meet the user request. For example, *get\_CGPA* procedure will have various queries to get the marks of student in each subject, calculate the total marks, and then decide the CGPA based on his total marks. This procedural query language tells the database what is required from the database and how to get them from the database. Relational algebra is a procedural query language.
- ii) **Non-Procedural Query Language:** Non-procedural queries will have single query on one or more tables to get result from the database. For example, get the name and address of the student with particular ID will have single query on STUDENT table. Relational Calculus is a non procedural language which informs what to do with the tables, but doesn't inform how to accomplish this.

These query languages basically will have queries on tables in the database. In the relational database, a table is known as relation. Records / rows of the table are referred as tuples. Columns of the table are also known as attributes. All these names are used interchangeably in relational database.

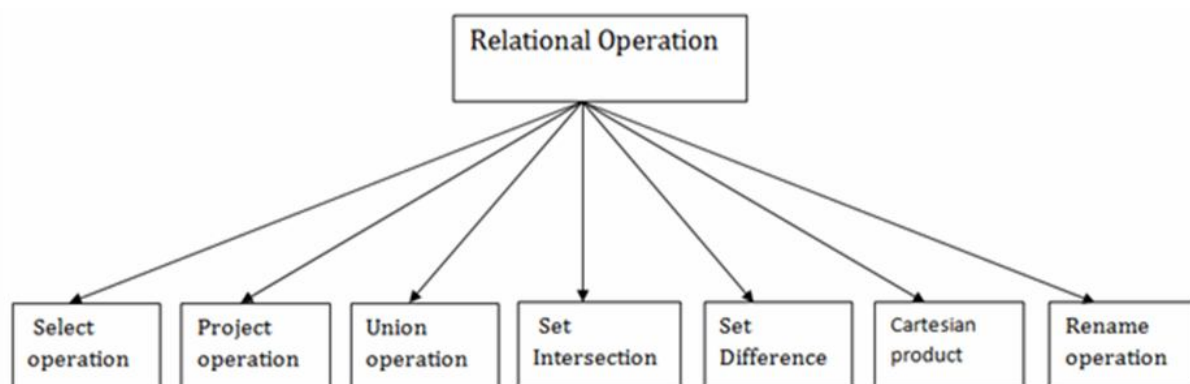
**Q43. What is relational algebra? List and explain various types of relational operations.**

*Ans :*

(Imp.)

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

**Types of Relational operation**



## 1. Select Operation

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma ( $\sigma$ ).

### i) Notation: $\sigma_p(r)$

**Where:**

$\sigma$  is used for selection prediction

$r$  is used for relation

$p$  is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like  $=, \neq, \geq, <, >, \leq$ .

**For example: LOAN Relation**

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

**Input**

- ii)  $\sigma_{\text{BRANCH\_NAME} = \text{"perryride"}}(\text{LOAN})$

**Output**

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

## 2. Project Operation

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by  $\Pi$ .

1. Notation:  $\Pi A_1, A_2, A_n(r)$

**Where**

$A_1, A_2, A_3$  is used as an attribute name of relation  $r$ .

**Example: CUSTOMER RELATION**

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

**Input**

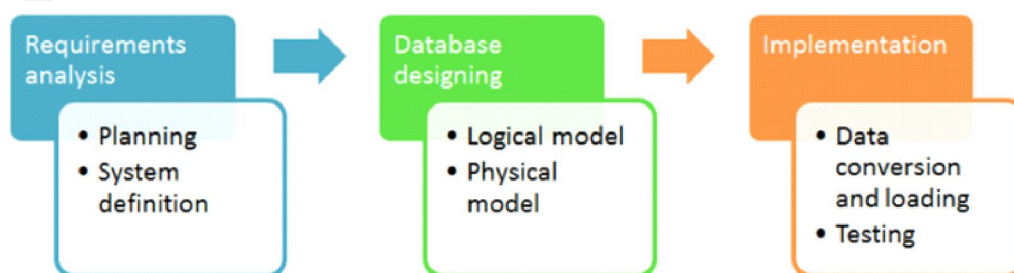
1.  $\Pi$  NAME, CITY (CUSTOMER)

**Output**

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

**3. Union Operation**

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by  $\cup$ .

**Notation:**  $R \cup S$ 

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

**Example****DEPOSITOR RELATION**

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

**BORROW RELATION**

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

**Input**

1.  $\Pi$  CUSTOMER\_NAME (BORROW)  $\cup$   $\Pi$  CUSTOMER\_NAME (DEPOSITOR)

**Output**

CUSTOMER_NAME	
Johnson	
Smith	
Hayes	
Turner	
Jones	
Lindsay	
Jackson	
Curry	
Williams	
Mayes	

**4. Set Intersection**

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R and S.
- It is denoted by intersection  $\cap$ .

i) **Notation:**  $R \cap S$

**Example:** Using the above DEPOSITOR table and BORROW table

**Input**

$\Pi \text{ CUSTOMER\_NAME (BORROW)} \cap \Pi \text{ CUSTOMER\_NAME (DEPOSITOR)}$

**Output**

CUSTOMER_NAME	
Smith	
Jones	

**5. Set Difference**

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus ( $-$ ).

ii) **Notation:**  $R - S$

**Example:** Using the above DEPOSITOR table and BORROW table

**Input**

$\Pi \text{ CUSTOMER\_NAME (BORROW)} - \Pi \text{ CUSTOMER\_NAME (DEPOSITOR)}$

**Output**

CUSTOMER_NAME	
Jackson	
Hayes	
Willians	
Curry	

**6. Cartesian product**

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by  $\times$ .

iii) Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

1. EMPLOYEE X DEPARTMENT

Output

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

## 7. Rename Operation

The rename operation is used to rename the output relation. It is denoted by **rho** ( $\rho$ ).

**Example:** We can use the rename operator to rename STUDENT relation to STUDENT1.

1.  $\rho(\text{STUDENT1}, \text{STUDENT})$

## Short Question and Answers

### 1. What is Database?

*Ans :*

A **database** is an organized collection of data, so that it can be easily accessed and managed.

**Database handlers** create a database in such a way that only one set of software program provides access of data to all the users.

The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many **dynamic websites** on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.

There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

**SQL** or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

### 2. Define DBMS.

*Ans :*

Database management System is software which is used to store and retrieve the database. For example, Oracle, MySQL, etc.; these are some popular DBMS tools.

- DBMS provides the interface to perform the various operations like creation, deletion, modification, etc.
- DBMS allows the user to create their databases as per their requirement.
- DBMS accepts the request from the application and provides specific data through the operating system.
- DBMS contains the group of programs which acts according to the user instruction.
- It provides security to the database.

### 3. What is the purpose of DBMS ?

*Ans :*

The Database Management System (DBMS) is defined as a software system that allows the user to define, create and maintain the database and provide control access to the data.

It is a collection of programs used for managing data and simultaneously it supports different types of users to create, manage, retrieve, update and store information.

#### Purpose

The purpose of DBMS is to transform the following:

- Data into information.
- Information into knowledge.
- Knowledge to the action.

The diagram given below explains the process as to how the transformation of data to information to knowledge to action happens respectively in the DBMS:

Previously, the database applications were built directly on top of the file system.

### 4. Data Definition Language (DDL)

*Ans :*

- DDL stands for Data Definition Language. It is used to define database structure or pattern.
  - It is used to create schema, tables, indexes, constraints, etc. in the database.
  - Using the DDL statements, you can create the skeleton of the database.
  - Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.
- Here are some tasks that come under DDL:
- **Create:** It is used to create objects in the database.

- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

### 5. Data Manipulation Language (DML)

*Ans :*

DML stands for Data Manipulation Language. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

### 6. Data Control Language

*Ans :*

- DCL stands for data control language. It is used to retrieve the stored or saved data.
  - The DCL execution is transactional. It also has rollback parameters.
- (But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

### 7. Transaction Control Language

*Ans :*

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

### 8. What is RDBMS ?

*Ans :*

RDBMS stands for Relational Database Management Systems;

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, MySQL and Microsoft Access are based on RDBMS.

It is called Relational Data Base Management System (RDBMS) because it is based on relational model introduced by E.F. Codd.

Data is represented in terms of tuples (rows) in RDBMS. Relational database is most commonly used database. It contains number of tables and each table has its own primary key.

Due to a collection of organized set of tables, data can be accessed easily in RDBMS.

### 9. What is database design?

*Ans :*

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and



maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

#### 10. Define the term transaction.

*Ans :*

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs. 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

##### A's Account

```
Open_Account(A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account(A)
```

##### B's Account

```
Open_Account(B)
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account(B)
```

#### 11. Explain ACID Properties and how they are useful to transactions?

*Ans :*

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability " commonly known as ACID properties " in order to ensure accuracy, completeness, and data integrity.

- (i) **Atomicity:** This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
- (ii) **Consistency:** The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
- (iii) **Durability:** The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
- (iv) **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.

#### 12. Who is database administrator?

*Ans :*

The life cycle of a database starts from designing, implementing to the administration of it. A database for any kind of requirement needs to

be designed perfectly so that it should work without any issues. Once all the design is complete, it needs to be installed. Once this step is complete, users start using the database. The database grows as the data grows in the database. When the database becomes huge, its performance comes down. Also accessing the data from the database becomes a challenge. There will be unused memory in the database, making the memory inevitably huge. This administration and maintenance of the database are taken care of by the database Administrator – DBA.

---

**13. What is relational model ?**

*Ans :*

Relational Model (RM) represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

---

**14. List out various types of Keys.**

*Ans :*

- i) **Primary Key:** A primary is a column or set of columns in a table that uniquely identifies tuples (rows) in that table.
- ii) **Super Key:** A super key is a set of one or more columns (attributes) to uniquely identify rows in a table.
- iii) **Candidate Key:** A super key with no redundant attribute is known as candidate key
- iv) **Alternate Key:** Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate or secondary keys.
- v) **Composite Key:** A key that consists of more than one attribute to uniquely identify rows (also known as records & tuples) in a table is called composite key.
- vi) **Foreign Key:** Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

---

**15. What is Key ?**

*Ans :*

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

### Choose the Correct Answers

1. \_\_\_\_\_ language used to query ad database [ a ]
  - a) Query
  - b) Relational
  - c) Structural
  - d) Compiler
2. The \_\_\_\_\_ operation allows the combining of two results [ b ]
  - a) Select
  - b) Join
  - c) Union
  - d) Intersection
3. Which of the following is not the degree of relationship [ a ]
  - a) Single
  - b) Binary
  - c) Ternary
  - d) n-ary
4. In an ER diagram double-rectangle represents [ b ]
  - a) Relationship
  - b) Weak entity set
  - c) Derived set
  - d) Attribute
5. Dis-advantage of file system is \_\_\_\_\_ [ d ]
  - a) Data redundancy
  - b) Accessing data
  - c) Data isolation
  - d) All
6. Which of following is a data model [ a ]
  - a) Entity - relational model
  - b) Relational data model
  - c) Object based data model
  - d) All
7. Which of the following is not binary operation. [ b ]
  - a) Union
  - b) Project
  - c) Set Difference
  - d) Product
8. Primary key must be \_\_\_\_\_ [ c ]
  - a) Not null
  - b) Unique
  - c) A or B
  - d) A and B
9. Which of the following is not an aggregation [ c ]
  - a) Max
  - b) Min
  - c) Select
  - d) Average
10. Which algebra is widely used in DBMS ? [ a ]
  - a) Relational Algebra
  - b) Arithmetic algebra
  - c) Both
  - d) None

## *Fill in the blanks*

1. \_\_\_\_\_ is a collection of related data is a collection of facts.
2. \_\_\_\_\_ are those who get benefits of DBMS.
3. A DBMS is a software for \_\_\_\_\_ and \_\_\_\_\_
4. The logical structure of the data stored in database in \_\_\_\_\_
5. \_\_\_\_\_ defines how file records are mapped onto disk blocks.
6. Controlling access to data and prevent unauthorised access and update is known as \_\_\_\_\_
7. Data independence is caterio into \_\_\_\_\_ and \_\_\_\_\_
8. The plan of database is known as \_\_\_\_\_
9. \_\_\_\_\_ is a series of database operations
10. \_\_\_\_\_ Symbol stands for selection predicate
11. \_\_\_\_\_ is used to combine the two or more results.
12. \_\_\_\_\_ Operator combines results of two select statements and return only those results belongs to first set.
13. The division operator is used when we have query which contain \_\_\_\_\_ keyword.

### Answer

1. Data Base
2. End users
3. Creating and Managing
4. Database Schema
5. File Organisations
6. Security of data
7. (Logical independence, Physical independence)
8. Schema
9. Transaction
10. ( $\sigma$ )
11. Union
12. Minus
13. All

## UNIT II

**Database Design and the E-R Model:** Overview of the Design Process, The EntityRelationship Model, Constraints, Removing Redundant Attributes in Entity Sets, EntityRelationship Diagrams, Reduction to Relational Schemas, Entity-Relationship Design Issues, Extended E-R Features, Alternative Notations for Modeling Data, Other Aspects of Database Design.

**Relational Database Design:** Features of Good Relational Designs, Atomic Domains and First Normal Form, Decomposition Using Functional Dependencies, Functional-Dependency Theory, Decomposition Using Multivalued Dependencies, Normal Forms- 2 NF, 3 NF, BCNF, The Database Design Methodology for Relational Databases.

### 2.1 DATABASE DESIGN MODEL AND ER – MODEL

#### 2.1.1 Overview of design process

##### Q1. What is database design?

*Ans :*

##### Meaning

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

##### Q2. What is a Good Database Design?

*Ans :*

A good database design process is governed by specific rules. The first rule is to avoid data

redundancy. It wastes space and increases the probability of faults and discrepancies within the database. The second rule is that the accuracy and comprehensiveness of information are imperative. A database containing erroneous information will lead to inaccurate analysis and reporting. Consequently, it can mislead decision-makers and adversely affect a company's performance. Therefore, it's important to keep things rules in mind when designing the database for your organization.

A well-designed database is the one that:

- Distributes your data into tables based on specific subject areas to decrease data redundancy
- Delivers the database the information needed to link the data in the tables
- Provides support, and guarantees the precision and reliability of data
- Caters to your information processing and reporting requirements
- Functions interactively with the database operators.

##### Q3. Why database design is important ?

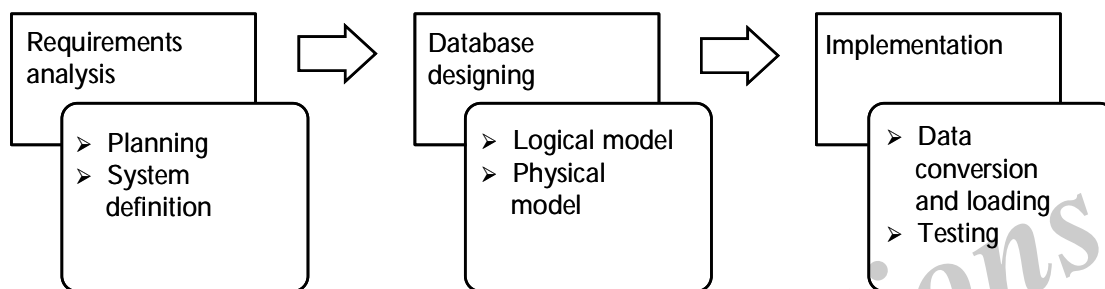
*Ans :*

The important consideration that can be taken into account while emphasizing the importance of database design can be explained in terms of the following points given below.

1. Database designs provide the blueprints of how the data is going to be stored in a system. A proper design of a database highly affects the overall performance of any application.

2. The designing principles defined for a database give a clear idea of the behavior of any application and how the requests are processed.
3. Another instance to emphasize the database design is that a proper database design meets all the requirements of users.
4. Lastly, the processing time of an application is greatly reduced if the constraints of designing a highly efficient database are properly implemented.

### Life Cycle



Although, the life cycle of a database is not an important discussion that has to be taken forward in this article because we are focused on the database design. But, before jumping directly on the designing models constituting database design it is important to understand the overall workflow and life-cycle of the database.

#### 1. Requirement Analysis

First of all, the planning has to be done on what are the basic requirements of the project under which the design of the database has to be taken forward. Thus, they can be defined as:-

- i) **Planning:** This stage is concerned with planning the entire DDLC (Database Development Life Cycle). The strategic considerations are taken into account before proceeding.
- ii) **System definition:** This stage covers the boundaries and scopes of the proper database after planning.

#### 2. Database Designing

The next step involves designing the database considering the user-based requirements and splitting them out into various models so that load or heavy dependencies on a single aspect are not imposed. Therefore, there has been some model-centric approach and that's where logical and physical models play a crucial role.

- i) **Physical Model:** The physical model is concerned with the practices and implementations of the logical model.
- ii) **Logical Model:** This stage is primarily concerned with developing a model based on the proposed requirements. The entire model is designed on paper without any implementation or adopting DBMS considerations.

### 3. Implementation

The last step covers the implementation methods and checking out the behavior that matches our requirements. It is ensured with continuous integration testing of the database with different data sets and conversion of data into machine understandable language. The manipulation of data is primarily focused on these steps where queries are made to run and check if the application is designed satisfactorily or not.

- i) **Data conversion and loading:** This section is used to import and convert data from the old to the new system.
- ii) **Testing:** This stage is concerned with error identification in the newly implemented system. Testing is a crucial step because it checks the database directly and compares the requirement specifications.

#### Q4. Explain database designing process ?

*Ans :*

(Imp.)

The process of designing a database carries various conceptual approaches that are needed to be kept in mind. An ideal and well-structured database design must be able to:

- Save disk space by eliminating redundant data.
- Maintains data integrity and accuracy.
- Provides data access in useful ways.
- Comparing Logical and Physical data models.

#### 1. Logical

A logical data model generally describes the data in as many details as possible, without having to be concerned about the physical implementations in the database. Features of logical data model might include:

- All the entities and relationships amongst them.
- Each entity has well-specified attributes.
- The primary key for each entity is specified.

- Foreign keys which are used to identify a relationship between different entities are specified.

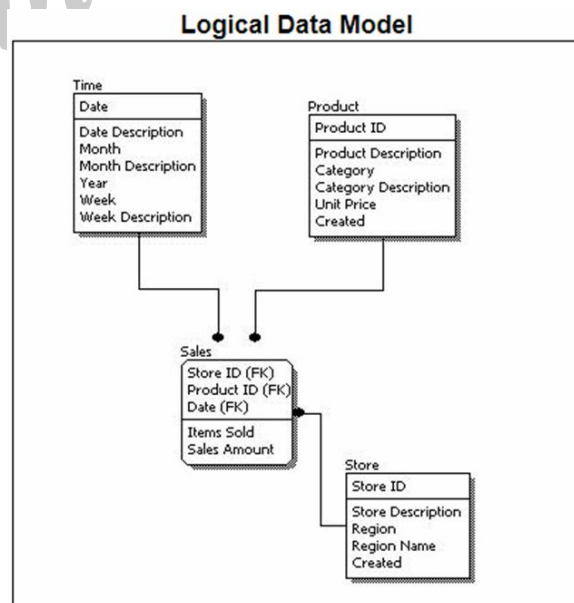
- Normalization occurs at this level.

A logical model can be designed using the following approach:

- Specify all the entities with primary keys.
- Specify concurrent relationships between different entities.
- Figure out each entity attributes
- Resolve many-to-many relationships.
- Carry out the process of normalization.

Also, one important factor after following the above approach is to critically examine the design based on requirement gathering. If the above steps are strictly followed, there are chances of creating a highly efficient database design that follows the native approach.

To understand these points, see the image below to get a clear picture.



If we compare the logical data model as shown in the figure above with some sample data in the diagram, we can come up with facts that in a conceptual data model there are no presence of a primary key whereas a logical data model has primary keys for all of its attributes. Also, logical data

model the cover relationship between different entities and carries room for foreign keys to establish relationships among them.

## 2. Physical

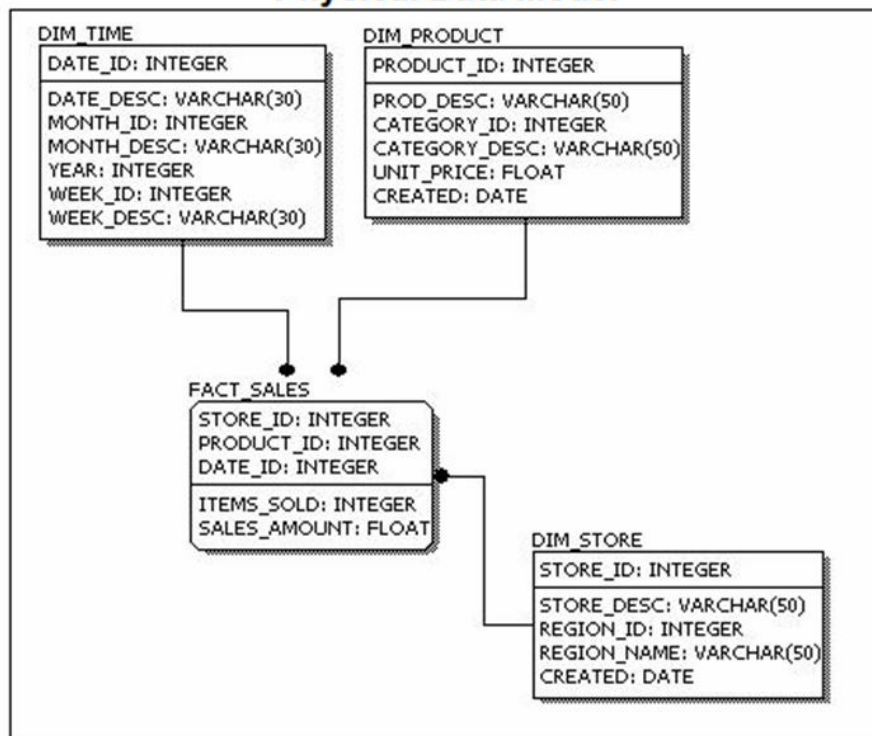
A Physical data model generally represents how the approach or concept of designing the database. The main purpose of the physical data model is to show all the structures of the table including the column name, column data type, constraints, keys (primary and foreign), and the relationship among tables. The following are the features of a physical data model:

- Specifies all the columns and tables.
- Specifies foreign keys that usually define the relationship between tables.
- Based on user requirements, de-normalization might occur.
- Since the physical consideration is taken into account so there will straightforward reasons for difference than a logical model.
- Physical models might be different for different RDBMS. For example, the data type column may be different in MySQL and SQL Server.

While designing a physical data model, the following points should be taken into consideration:

- i) Convert the entities into tables.
- ii) Convert the defined relationships into foreign keys.
- iii) Convert the data attributes into columns.
- iv) Modify the data model constraints based on physical requirements.

### Physical Data Model





Comparing this physical data model with the logical with the previous logical model, we might conclude the differences that in a physical database entity names are considered table names and attributes are considered column names. Also, the data type of each column is defined in the physical model depending on the actual database used.

### 2.1.2 The Entity Relationship model

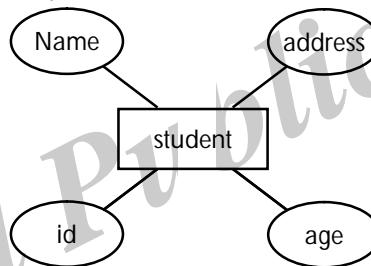
#### Q5. What is ER Model ?

*Ans :*

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

#### For example

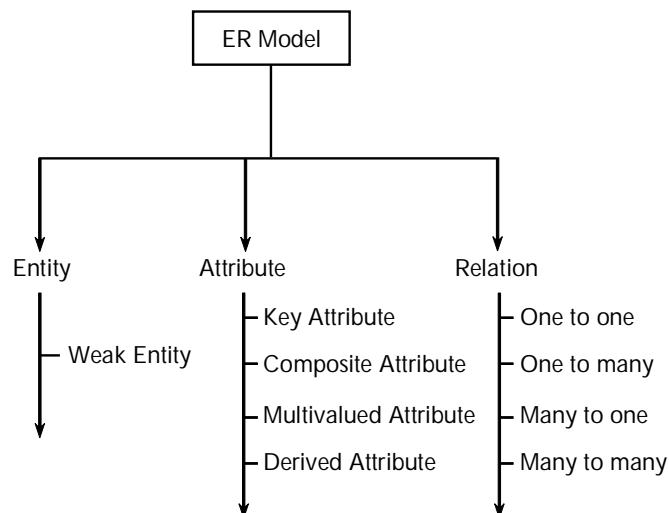
Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



#### Q6. List and explain various components of ER Model ?

*Ans :*

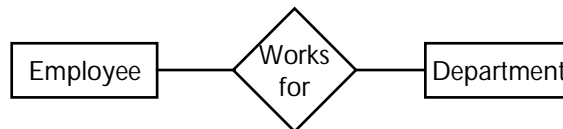
#### Component of ER Diagram



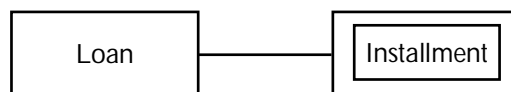
## 1. Entity

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



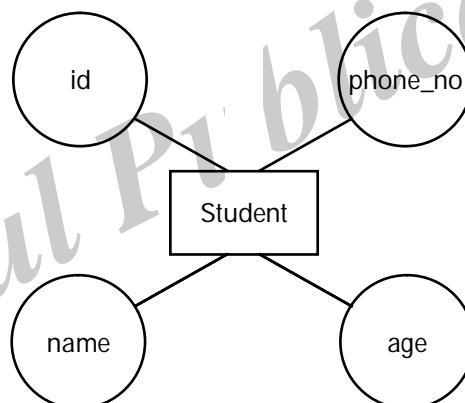
(a) **Weak Entity:** An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



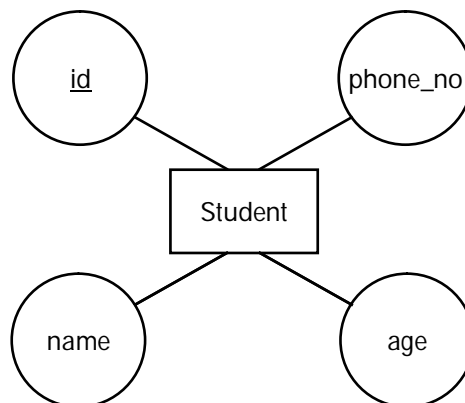
## 2. Attribute

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

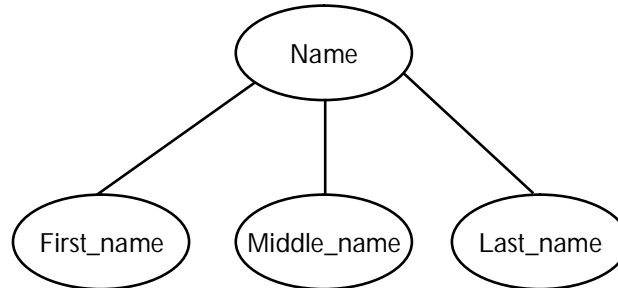
**For example,** id, age, contact number, name, etc. can be attributes of a student.



(b) **Key Attribute:** The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.

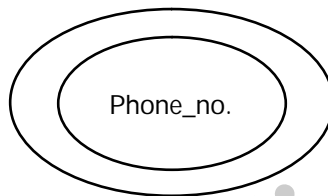


- (c) **Composite Attribute:** An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



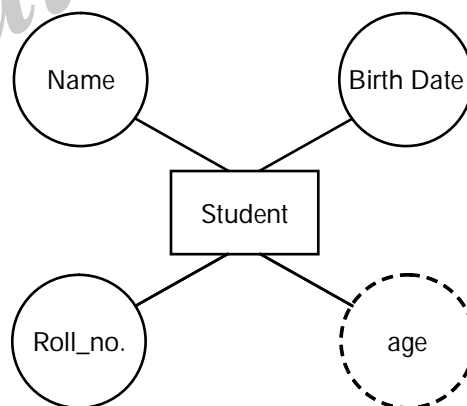
- (d) **Multivalued Attribute:** An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

**For example,** a student can have more than one phone number.



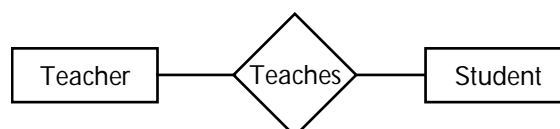
- (e) **Derived Attribute:** An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

**For example,** A person's age changes over time and can be derived from another attribute like Date of birth.



### 3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

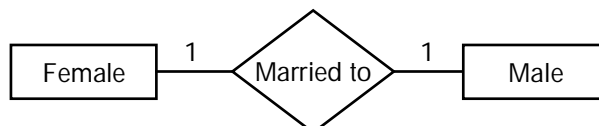


Types of relationship are as follows:

**(a) One-to-One Relationship**

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

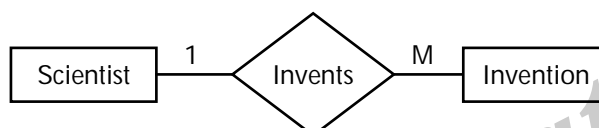
**For example,** A female can marry to one male, and a male can marry to one female.



**(b) One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

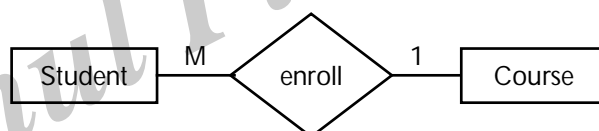
**For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



**(c) Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

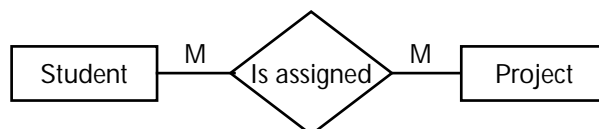
**For example,** Student enrolls for only one course, but a course can have many students.



**(d) Many-to-many relationship**

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

**For example,** Employee can assign by many projects and project can have many employees.



**Q7. What are the differences between strong entity set and weak entity set.**

*Ans :*

**(Imp.)**

**i) Strong Entity**

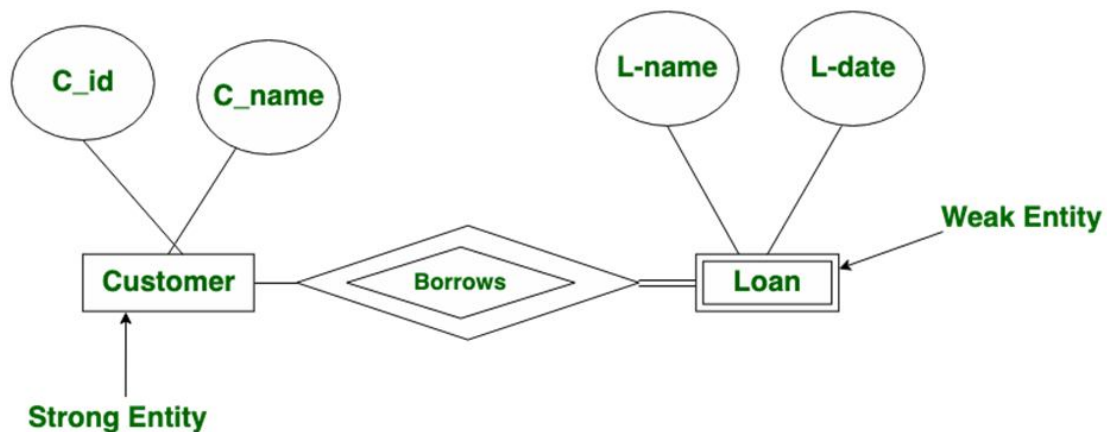
A strong entity is not dependent on any other entity in the schema. A strong entity will always have a primary key. Strong entities are represented by a single rectangle. The relationship of two strong entities is represented by a single diamond.

Various strong entities, when combined together, create a strong entity set.

**ii) Weak Entity**

A weak entity is dependent on a strong entity to ensure its existence. Unlike a strong entity, a weak entity does not have any primary key. It instead has a partial discriminator key. A weak entity is represented by a double rectangle.

The relation between one strong and one weak entity is represented by a double diamond.



S.No.	Strong Entity Set	Weak Entity Set
1.	Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
2.	It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
3.	It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
4.	The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
5.	Primary Key is one of its attributes which helps to identify its member. entity set.	In a weak entity set, it is a combination of primary key and partial key of the strong
6.	In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
7.	The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

**2.1.3 Constraints****Q8. What is relational integrity constraints in DBMS ?**

*Ans :*

**(Imp.)**

On modeling the design of the relational database we can put some restrictions like what values are allowed to be inserted in the relation, what kind of modifications and deletions are allowed in the relation. These are the restrictions we impose on the relational database.

In models like ER models, we did not have such features.

Constraints in the databases can be categorized into 3 main categories:

1. Constraints that are applied in the data model is called Implicit constraints.
2. Constraints that are directly applied in the schemas of the data model, by specifying them in the DDL(Data Definition Language). These are called as schema-based constraints or Explicit constraints.
3. Constraints that cannot be directly applied in the schemas of the data model. We call these Application based or semantic constraints.

So here we will deal with Implicit constraints.

Mainly Constraints on the relational database are of 4 types:

1. Domain constraints
2. Key constraints
3. Entity Integrity constraints
4. Referential integrity constraints

Let discuss each of the above constraints in detail.

### 1. Domain constraints

- i) Every domain must contain atomic values(smallest indivisible units) it means composite and multi-valued attributes are not allowed.
- ii) We perform datatype check here, which means when we assign a data type to a column we limit the values that it can contain. Eg. If we assign the datatype of attribute age as int, we cant give it values other then int datatype.

### 2. Key constraints or uniqueness constraints

- i) These are called uniqueness constraints since it ensures that every tuple in the relation should be unique.
- ii) A relation can have multiple keys or candidate keys (minimal superkey), out of which we choose one of the keys as

primary key, we don't have any restriction on choosing the primary key out of candidate keys, but it is suggested to go with the candidate key with less number of attributes.

- iii) Null values are not allowed in the primary key, hence Not Null constraint is also a part of key constraint.

### 3. Entity Integrity Constraints

Entity Integrity constraints says that no primary key can take NULL value, since using primary key we identify each tuple uniquely in a relation.

### 4. Referential Integrity Constraints

- i) The Referential integrity constraints is specified between two relations or tables and used to maintain the consistency among the tuples in two relations.
- ii) This constraint is enforced through foreign key, when an attribute in the foreign key of relation R1 have the same domain(s) as the primary key of relation R2, then the foreign key of R1 is said to reference or refer to the primary key of relation R2.
- iii) The values of the foreign key in a tuple of relation R1 can either take the values of the primary key for some tuple in relation R2, or can take NULL values, but can't be empty.

#### 2.1.4 Removing redundant attributes in entity set

#### Q9. What is Data Redundancy?

*Ans :*

Data redundancy occurs in database systems which have a field that is repeated in two or more tables. When customer data is duplicated and attached with each product bought, then redundancy of data is a known source of inconsistency, since the entity "customer" might appear with different values for a given attribute.

Data redundancy leads to data anomalies and corruption and should be avoided when creating a relational database consisting of several entities.

Database normalization prevents redundancy and makes the best possible usage of storage. The proper use of foreign keys can minimize data redundancy and reduce the chance of destructive anomalies appearing. Concerns with respect to the efficiency and convenience can sometimes result in redundant data design despite the risk of corrupting the data.

**Q10. How does data redundancy occur?**

*Ans :*

Data redundancy can be designed; for example, suppose you want to back up your company's data nightly. This creates a redundancy. Data redundancy can also occur by mistake. For example, the database designer who created a system with a new record for each sale may not have realized that his design caused the same address to be entered repeatedly. You may also end up with redundant data when you store the same information in multiple systems. For instance, suppose you store the same basic employee information in Human Resources records and in records maintained for your local site office.

**Q11. List out advantages and disadvantages of data redundancy.**

*Ans :*

(Imp.)

**Advantages**

Although data redundancy sounds like a negative event, there are many organizations that can benefit from this process when it's intentionally built into daily operations.

**1. Alternative data backup method**

Backing up data involves creating compressed and encrypted versions of data and storing it in a computer system or the cloud. Data redundancy offers an extra layer of protection and reinforces the backup by replicating data to an additional system. It's often an advantage when companies incorporate data redundancy into their disaster recovery plans.

**2. Better data security**

Data security relates to protecting data, in a database or a file storage system, from unwanted activities such as cyberattacks or

data breaches. Having the same data stored in two or more separate places can protect an organization in the event of a cyberattack or breach — an event which can result in lost time and money, as well as a damaged reputation.

**3. Faster data access and updates**

When data is redundant, employees enjoy fast access and quick updates because the necessary information is available on multiple systems. This is particularly important for customer service-based organizations whose customers expect promptness and efficiency.

**4. Improved data reliability**

Data that is reliable is complete and accurate. Organizations can use data redundancy to double check data and confirm it's correct and completed in full - a necessity when interacting with customers, vendors, internal staff, and others.

**Disadvantages**

Although there are noteworthy advantages of intentional data redundancy, there are also several significant drawbacks when organizations are unaware of its presence.

**1. Possible data inconsistency**

Data redundancy occurs when the same piece of data exists in multiple places, whereas data inconsistency is when the same data exists in different formats in multiple tables. Unfortunately, data redundancy can cause data inconsistency, which can provide a company with unreliable and/or meaningless information.

**2. Increase in data corruption**

Data corruption is when data becomes damaged as a result of errors in writing, reading, storage, or processing. When the same data fields are repeated in a database or file storage system, data corruption arises. If a file gets corrupted, for example, and an employee tries to open it, they may get an error message and not be able to complete their task.

### 3. Increase in database size

Data redundancy may increase the size and complexity of a database - making it more of a challenge to maintain. A larger database can also lead to longer load times and a great deal of headaches and frustrations for employees as they'll need to spend more time completing daily tasks.

### 4. Increase in cost

When more data is created due to data redundancy, storage costs suddenly increase. This can be a serious issue for organizations who are trying to keep costs low in order to increase profits and meet their goals. In addition, implementing a database system can become more expensive.

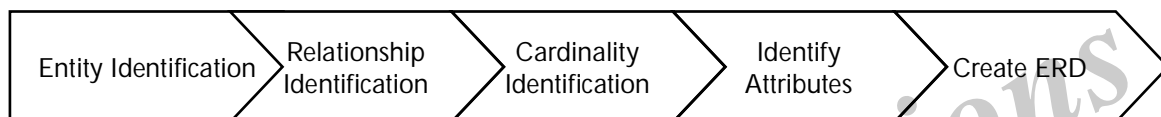
## 2.1.5 Entity Relationship Diagram

### Q12. How to create an ER Diagram ?

Ans :

(Imp.)

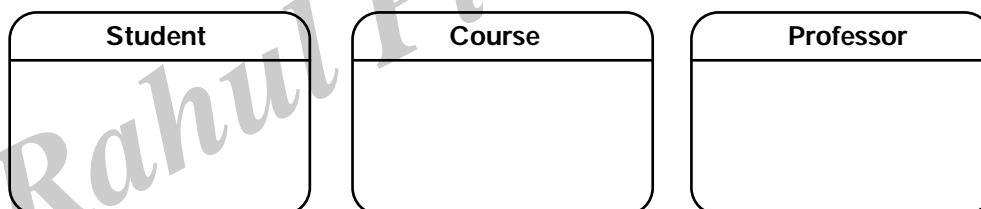
Following are the steps involved to develop the ER Diagrams.



#### Step 1: Entity Identification

We have three entities

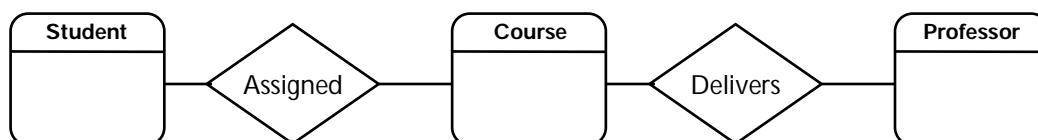
- Student
- Course
- Professor



#### Step 2: Relationship Identification

We have the following two relationships

- The student is assigned a course
- Professor delivers a course

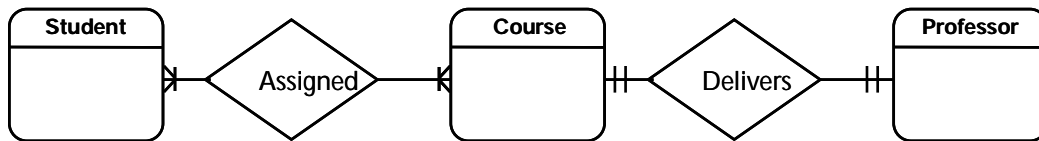


#### Step 3: Cardinality Identification

For them problem statement we know that,

- A student can be assigned **multiple** courses
- A Professor can deliver only **one** course





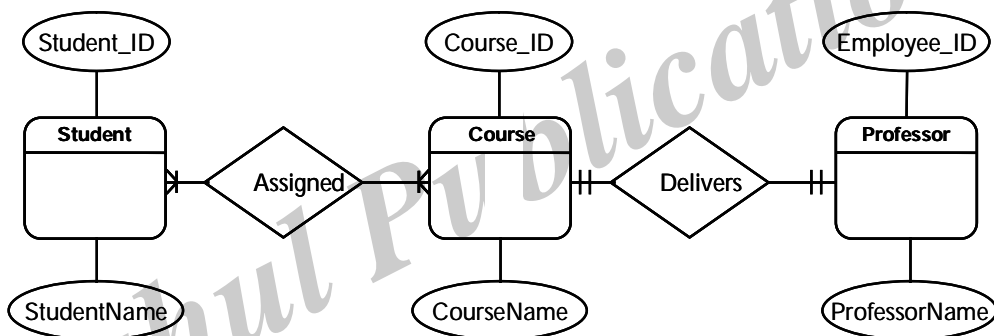
#### Step 4: Identify Attributes

You need to study the files, forms, reports, data currently maintained by the organization to identify attributes. You can also conduct interviews with various stakeholders to identify entities. Initially, it's important to identify the attributes without mapping them to a particular entity.

Once, you have a list of Attributes, you need to map them to the identified entities. Ensure an attribute is to be paired with exactly one entity. If you think an attribute should belong to more than one entity, use a modifier to make it unique.

Once the mapping is done, identify the primary Keys. If a unique key is not readily available, create one.

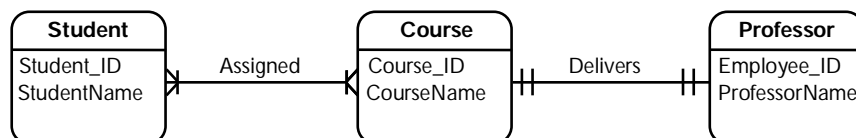
Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName



For Course Entity, attributes could be Duration, Credits, Assignments, etc. For the sake of ease we have considered just one attribute.

#### Step 5: Create the ERD Diagram

A more modern representation of Entity Relationship Diagram Example



#### Q13. List out best practices for developing effective ER Diagrams ?

Ans :

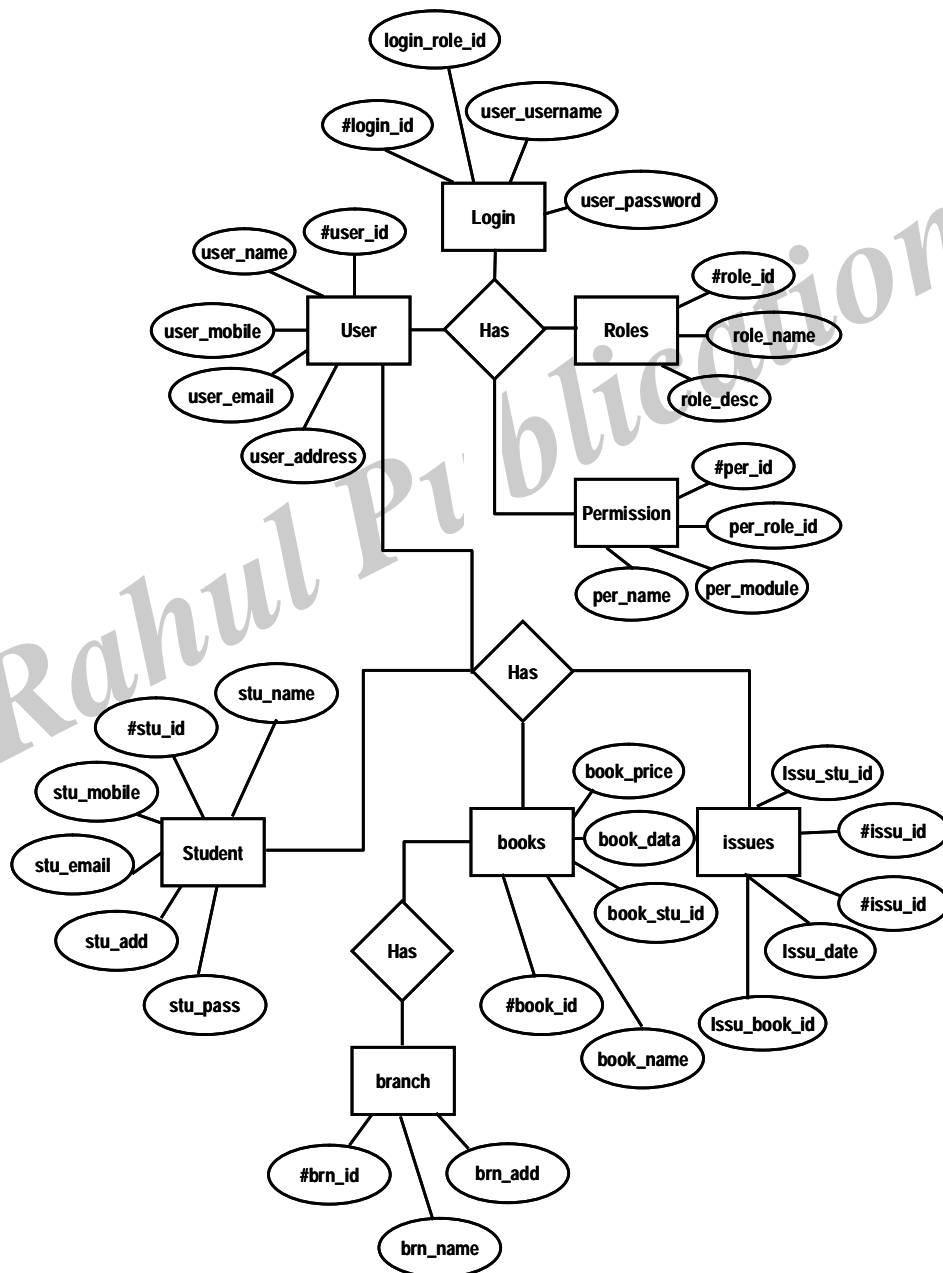
Here are some best practice or example for Developing Effective ER Diagrams.

- Eliminate any redundant entities or relationships.
- You need to make sure that all your entities and relationships are properly labeled.

- There may be various valid approaches to an ER diagram. You need to make sure that the ER diagram supports all the data you need to store
- You should assure that each entity only appears a single time in the ER diagram
- Name every relationship, entity, and attribute are represented on your diagram
- Never connect relationships to each other
- You should use colors to highlight important portions of the ER diagram

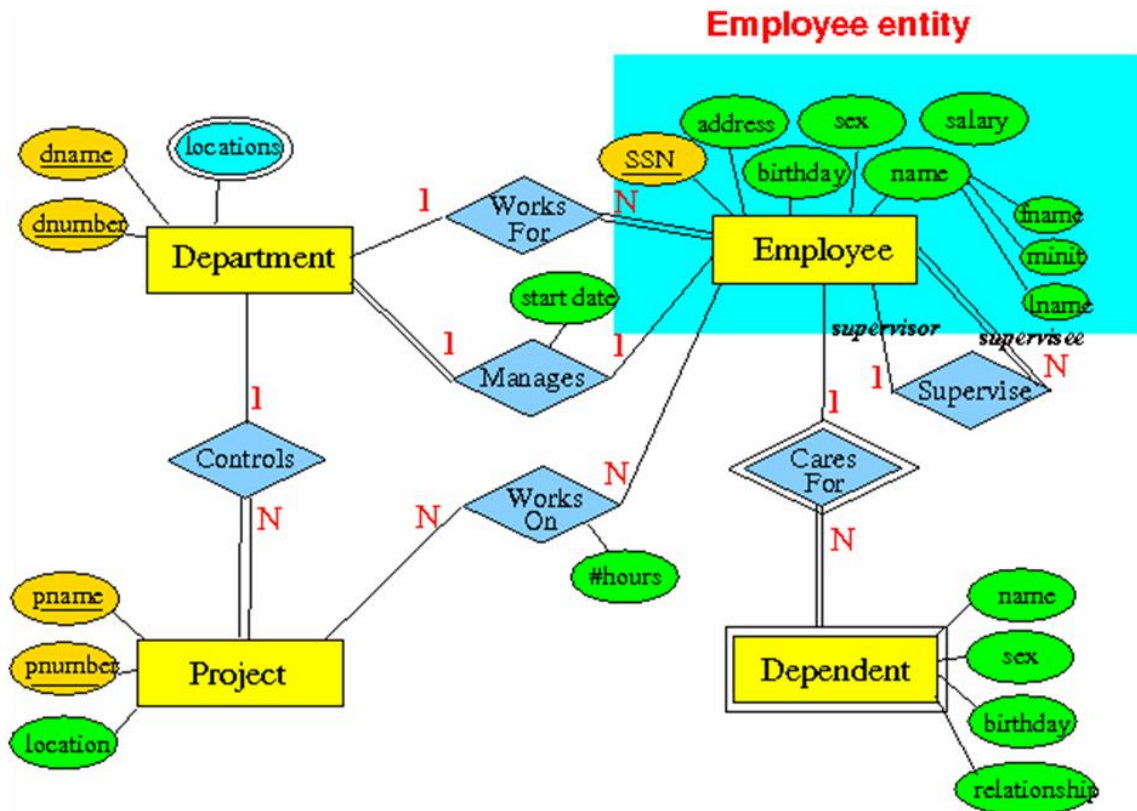
**Q14. Draw an ER Diagram for college management system.**

*Ans. :*



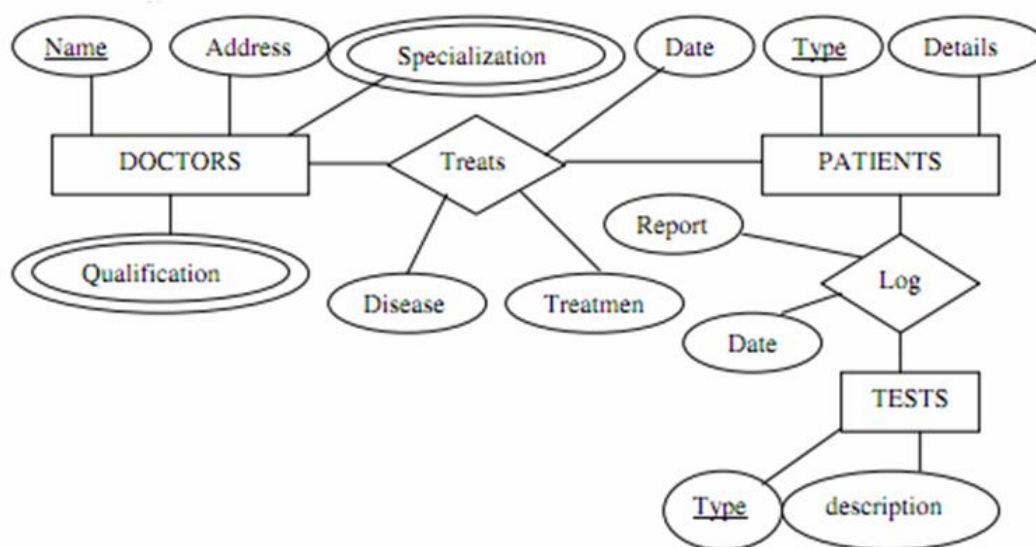
Q15. Draw an ER Diagram for describing employee information.

Ans :



Q16. Draw an ER Diagram for Hospital information.

Ans :



**Q17. List out the benefits of ER Diagrams***Ans :***Benefits of ER diagrams**

ER diagrams constitute a very useful framework for creating and manipulating databases. First, ER diagrams are easy to understand and do not require a person to undergo extensive training to be able to work with it efficiently and accurately. This means that designers can use ER diagrams to easily communicate with developers, customers, and end users, regardless of their IT proficiency.

Second, ER diagrams are readily translatable into relational tables which can be used to quickly build databases. In addition, ER diagrams can directly be used by database developers as the blueprint for implementing data in specific software applications.

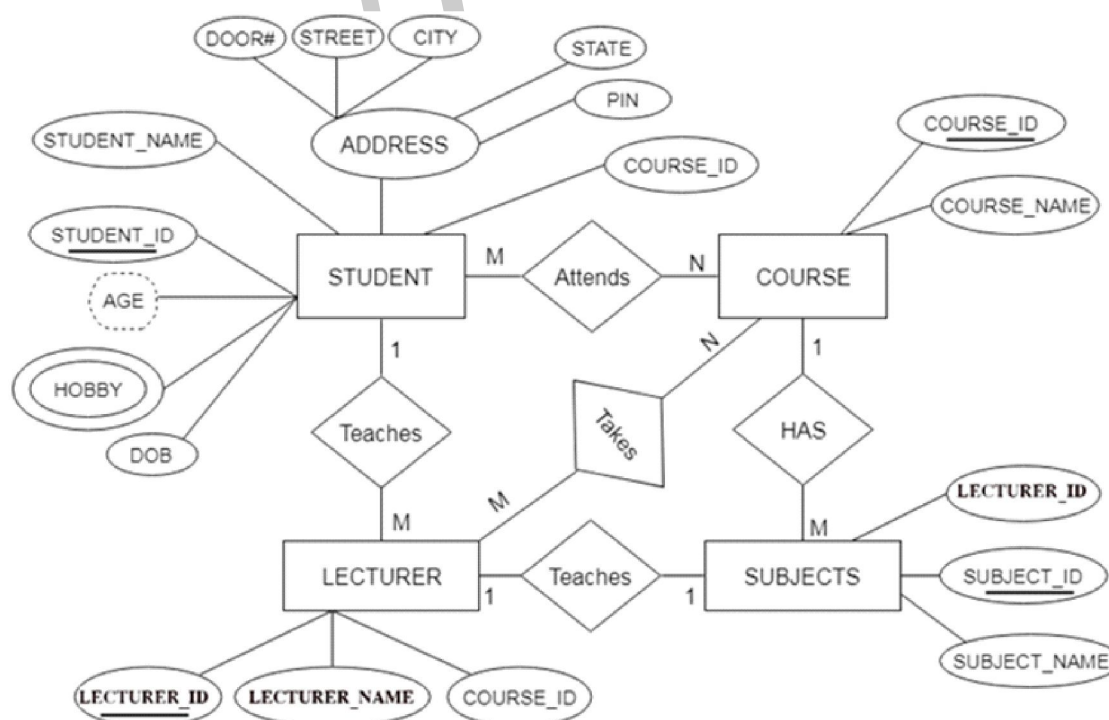
Lastly, ER diagrams may be applied in other contexts such as describing the different relationships and operations within an organization.

**2.1.6 Reduction to relational schemas****Q18. Explain the process of creating ER Diagram into table with an example.***Ans :***(Imp.)****Reduction of ER diagram to Table**

The database can be represented using the notations, and these notations can be reduced to a collection of tables.

In the database, every entity set or relationship set can be represented in tabular form.

The ER diagram is given below



There are some points for converting the ER diagram to the table:

- **Entity type becomes a table:** In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.
- **All single-valued attribute becomes a column for the table:** In the STUDENT entity, STUDENT\_NAME and STUDENT\_ID form the column of STUDENT table. Similarly, COURSE\_NAME and COURSE\_ID form the column of COURSE table and so on.
- **A key attribute of the entity type represented by the primary key:** In the given ER diagram, COURSE\_ID, STUDENT\_ID, SUBJECT\_ID, and LECTURE\_ID are the key attribute of the entity.
- **The multivalued attribute is represented by a separate table:** In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD\_HOBBY with column name STUDENT\_ID and HOBBY. Using both the column, we create a composite key.
- **Composite attribute represented by components:** In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.
- **Derived attributes are not considered in the table:** In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:

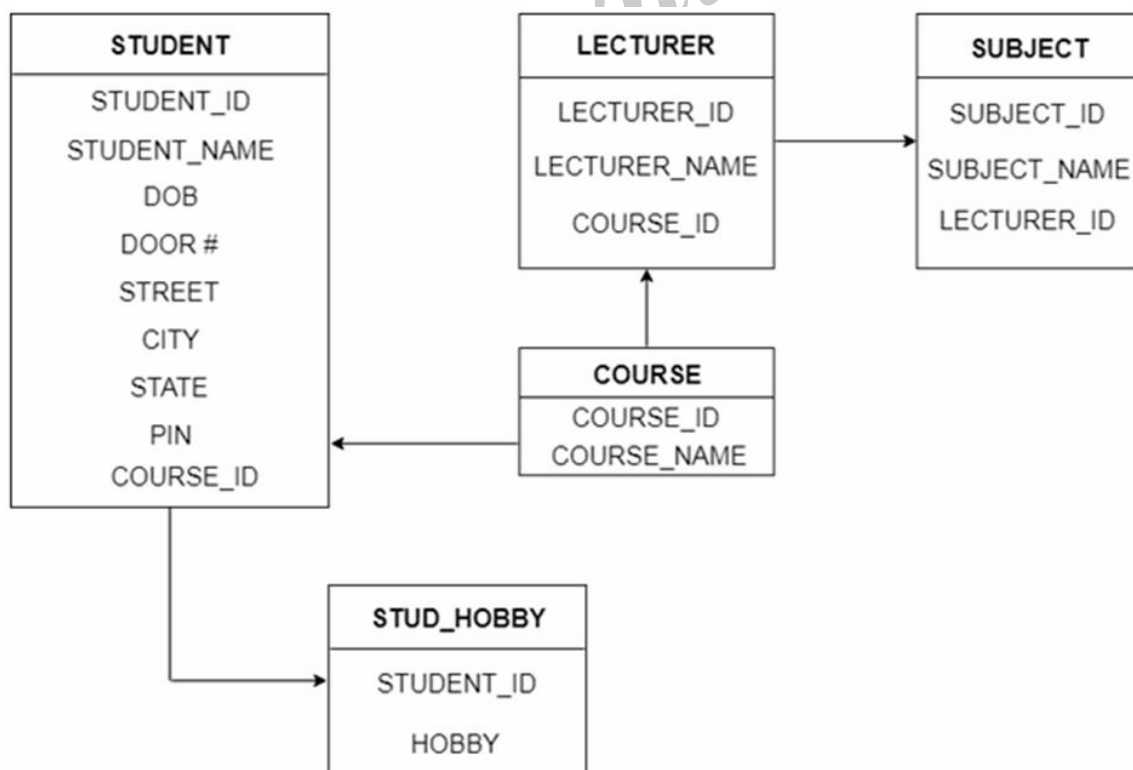
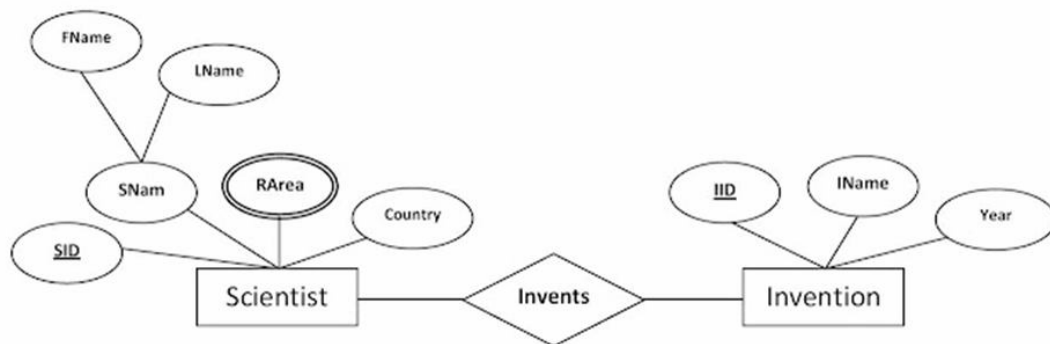


Fig.: Table structure

Q19. Convert/reduce the ER Diagram given in figure below.



Ans :

Given in the figure;

#### Entity sets and relationship sets

Name	Entity set / Relationship set	Type
Scientist	Entity set	Strong entity set
Invention	Entity set	Strong entity set
Invents	Relationship set	Many-to-Many relationship

#### Entity set **Scientist**

Attributes	Attribute Type	Description
SID	Simple and Primary key	Scientist ID
SName	Composite	Scientist Name
RArea	Multi-valued	Research Area
Country	Simple	Country

#### Entity set **Invention**

Attributes	Attribute Type	Description
IID	Simple and Primary key	Invention ID
IName	Simple	Name of the invention
Year	Simple	Year of invention

#### Reduction into relational schema

**Strong entity sets** – Entity set that has a primary key to uniquely represent each entity is Strong entity set.

Strong entity sets can be converted into relational schema by having the entity set name as the relation schema name and the attributes of that entity set as the attributes of relation schema.

Then we have,

Scientist (SID, SName, RArea, Country)

Invention (IID, IName, Year)

**1. After converting strong entity sets into relation schema**Scientist (SID, SName, RArea, Country)Invention (IID, IName, Year)**Composite attributes**

If an attribute can be further divided into two or more component attributes, that attribute is called composite attribute.

While converting into relation schemas, component attributes can be part of the strong entity sets' relation schema. No need to retain the composite attribute.

In our case, SName becomes FName, and LName as follows;

Scientist (SID, FName, LName, RArea, Country)**2. After converting composite attributes into relation schema**Scientist (SID, FName, LName, RArea, Country)Invention (IID, IName, Year)**Multi-valued attributes**

Attributes that may have multiple values are referred as multi-valued attributes.

In our ER diagram, RArea is a multi-valued attribute. That means, a scientist may have one or more areas as their research areas.

To reduce a multi-valued attribute into a relation schema, we have to create a separate table for each multi-valued attribute. Also, we need to include the primary key of strong entity set (parent entity set where the multi-valued attribute belongs) as a foreign key attribute to establish link.

In our case, the strong entity set Scientist will be further divided as follows;

Scientist (SID, FName, LName, RArea, Country)Scientist\_Area (SID, RArea)**3. After converting multi-valued attributes into relation schema**Scientist (SID, FName, LName, Country)Scientist\_Area (SID, RArea)Invention (IID, IName, Year)**Relationship set**

The association between two or more entity sets is termed as relationship set.

A relationship may be either converted into a separate table or not. That can be decided based on the type of the relationship. Only many-to-many relationship needs to be created as a separate table.

Here, we are given a many-to-many relationship. That means,

- One entity (record/row) of Scientist is related to one or more entities (records/rows) of Invention entity set (that is, one scientist may have one or more inventions) and,
- One entity (record/row) of Invention is related to one or more entities (records/rows) of Scientist entity set. (that is, one or more scientists may have invented one thing collectively).

To reduce the relationship invents into relational schema, we need to create a separate table for Invents, because Invents is a many-to-many relationship set. Hence, create a table Invents with the primary keys of participating entity sets (both, Scientist and Invention) as the attributes.

Then we have,

Invents (SID, IID)

Here, SID and IID are both foreign keys and collectively forms the primary key of Invents table.

Finally, we have the following relation schemas;

4. After converting relationship sets into relation schema
Scientist ( <u>SID</u> , FName, LName, Country)
Scientist_Area ( <u>SID</u> , RArea)Invention
( <u>IID</u> , IName, Year)
Invents ( <u>SID</u> , <u>IID</u> )

### 2.1.7 Entity Relationship Design Issues

**Q20. Discuss the basic design issues of an ER database schema ?**

*Ans :*

The basic design issues of an ER database schema in the following points:

- 1. Use of Entity Set vs Attributes:** The use of an entity set or attribute depends on the structure of the real-world enterprise that is being modelled and the semantics associated with its attributes. It leads to a mistake when the user use the primary key of an entity set as an attribute of another entity set. Instead, he should use the relationship to do so. Also, the primary key attributes are implicit in the relationship set, but we designate it in the relationship sets.
- 2. Use of Entity Set vs. Relationship Sets:** It is difficult to examine if an object can be best expressed by an entity set or relationship set. To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities. If there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.
- 3. Use of Binary vs n-ary Relationship Sets:** Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships. For example, we can create and represent a ternary relationship 'parent' that may relate to a child, his father, as well as his mother. Such relationship can also be represented by two binary relationships i.e, mother and father, that may relate to their child. Thus, it is possible to represent a non-binary relationship by a set of distinct binary relationships.
- 4. Placing Relationship Attributes:** The cardinality ratios can become an affective measure in the placement of the relationship attributes. So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set. The decision of placing the specified attribute as a relationship or entity attribute should possess the characteristics of the real world enterprise that is being modelled.

**For example,** if there is an entity which can be determined by the combination of participating entity sets, instead of determing it as a separate entity. Such type of attribute must be associated with the many-to-many relationship sets.



Thus, it requires the overall knowledge of each part that is involved in designing and modelling an ER diagram. The basic requirement is to analyse the real-world enterprise and the connectivity of one entity or attribute with other.

### 2.1.8 Extended ER Features

**Q21. What are the main features of Entity Relationship design issues?**

(OR)

**Discuss various extended ER features.**

*Ans :*

The various extended E-R features are as follows,

#### 1. Generalization

Generalization is a bottom-up approach which creates generalized entities. In this approach, entities present in the lower level combine together to form a higher level entity. Higher level entity is also called super class entity and lower level entity is also called subclass entity. Subclass/lower level entities are the entities that possess similar characteristics. These characteristics can be attributes, associations or methods. Generalization method aims at hiding the distinguishing characteristics of subclass entities thereby grouping all the similar characteristics.

#### 2. Specialization

Specialization is a top-down approach in which lower-level entities are formed by the decomposition of a higher level entity. Higher level entity is the super class entity and lower level entity is the subclass entity. This method identifies subsets of super class entity that share differentiating characteristics and separates those sets. These sets are known as sub-class sets. In specialization method, first a super class is defined followed by its sub-classes and then their attributes and relationships are defined.

#### 3. Aggregation

Aggregation represents a relationship between a whole object and its component parts.

Aggregation is the process of compiling information on an object, thereby abstracting a higher level object. In this manner, the entity person

is derived by aggregating the characteristics like name, address and social security number. Another form of aggregation is abstracting a relationship between objects and viewing the relationship as an object. In essence, the ENROLMENT relationship between entities student and course could be viewed as entity REGISTRATION.

In simple terms it can be said that "Aggregation is an abstraction through which relationships are treated as higher level entities". Aggregation allows the users to indicate that a relationship set (identified through a dashed box) participates in other relationship set. This is shown in the below figure with a dashed box around sponsors (and its participating entity sets) used to denote aggregation. This effectively allows the users to treat sponsors as an entity set for purposes of defining the monitors relationship set.

#### 4. Composition

Composition is derived from aggregation and is used to represent entity relationships having strong ownerships and coincidental lifetime among 'whole' and 'part' in a whole-part relationship. In composite aggregation, the part belongs to a single composite object. In other words, the composition is responsible for handling the creation and deletion of the parts. An example of composition is 'Publishes' relationship in a whole-part relationship where whole is 'Website' and part is 'Advertisement'. In such a relationship, the advertisement belongs to only one website?

#### 5. Attribute Inheritance

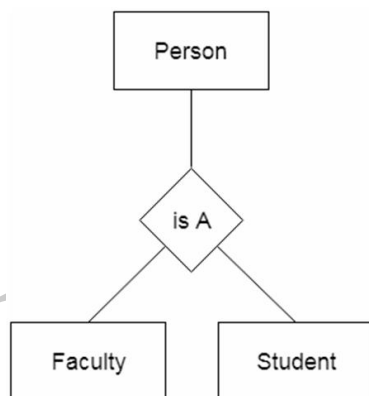
In database system, inheritance can be defined as a mechanism in which one class inherits the feature of another class. The class which acquires the features (attributes) are called subclass or child class whereas the class which gets inherited is called as a super class or parent class. A super class contains all the common variables in them.

Inheritance allows one class to specialize another class using addition and substitution mechanisms. Some systems support single inheritance while others support multiple inheritance. Single inheritance is the class specialization of only one super class while multiple inheritance allows a class to be the specialization of multiple classes.

**Q22. Explain the concept of Generalization.***Ans :*

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

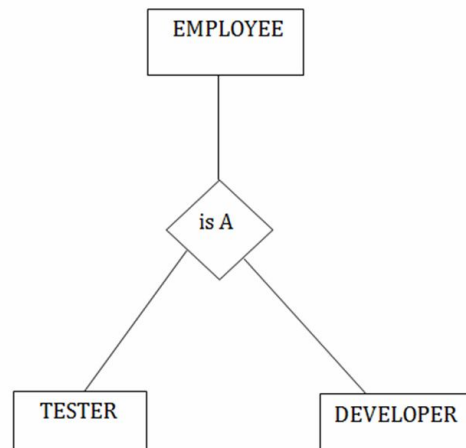
**For example,** Faculty and Student entities can be generalized and create a higher level entity Person.

**Q23. Explain Specialization in detail.***Ans :*

- Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

**For example**

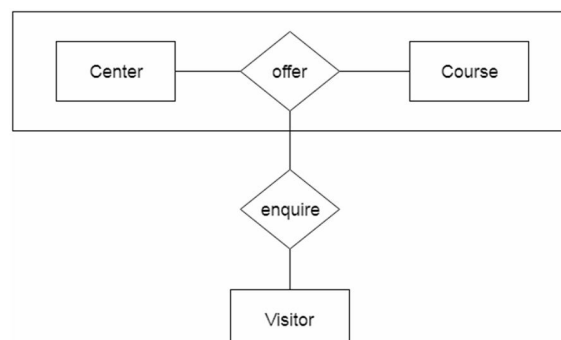
In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.

**Q24. Explain Aggregation in detail.***Ans :*

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

**For example**

Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



### 2.1.9 Alternative notations for modelling data

#### Q25. What are data modelling notations ?

*Ans :* (Imp.)

Following are the symbols( Notations ) used for constructing ER Diagrams.

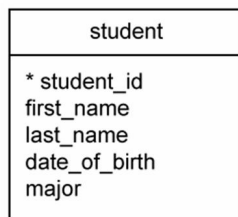
#### 1. Entities

**Definition:** An entity is a representation of a class of object. It can be a person, place, thing, etc. Entities usually have attributes that describe them.

In crow's foot notation, an entity is represented by a rectangle, with its name on the top. The name is singular (entity) rather than plural (entities).

#### 2. Attributes

**Definition:** An attribute is a property that describes a particular entity.



} attributes

The attribute(s) that uniquely distinguishes an instance of the entity is the **identifier**. Usually, this type of attribute is marked with an asterisk.

#### 3. Relationships

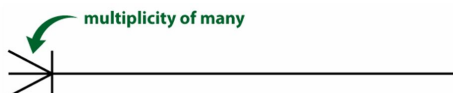
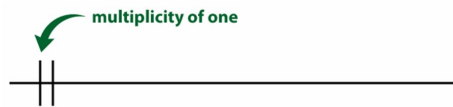
**Definition:** Relationships illustrate the association between two entities. They are presented as a straight line. Usually, each relationship has a name, expressed as a verb, written on the relationship line. This describes what kind of relationship connects the objects.

Note that the mentioned type of relationship is binary. In the Entity-Relationship model, representing a ternary or higher order of relationship is problematic.

#### 4. Cardinality

Relationships have two indicators. These are shown on both sides of the line.

- The first one (often called multiplicity) refers to the maximum number of times that an instance of one entity can be associated with instances in the related entity. It can be one or many.



- The second describes the minimum number of times one instance can be related to others. It can be zero or one, and accordingly describes the relationship as optional or mandatory.



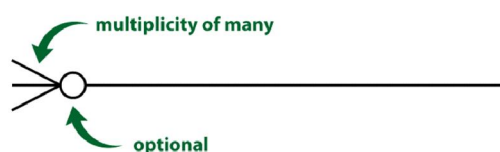
The combination of these two indicators is always in a specific order. Placed on the outside edge of the relationship, the symbol of multiplicity comes first. The symbol indicating whether the relationship is mandatory or optional is shown after the symbol of multiplicity.

#### In crow's foot notation:

- A multiplicity of one and a mandatory relationship is represented by a straight line perpendicular to the relationship line.
- A multiplicity of many is represented by the three-pronged 'crow-foot' symbol.
- An optional relationship is represented by an empty circle.

Finally, there are four possible edges to the relationship, illustrated here:

- zero or many



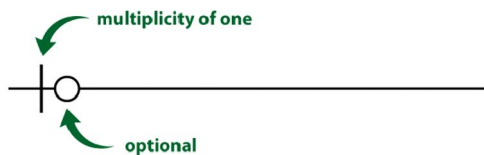
- one or many



- one and only one



- zero or one



Relationship degrees make them readable as:

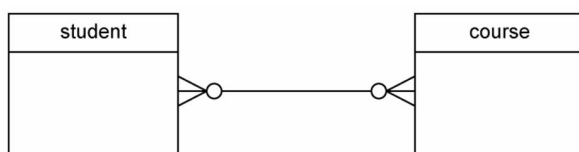
- One-to-one



- One-to-many



- Many-to-many



### 2.1.10 Other aspects of database design

**Q26. What are the other aspects of database design ?**

*Ans :* (Imp.)

Following are the aspects of good database design

#### i) Defining the System Parameters

Ideally, every project should begin with a clear definition of what the scope of the project, the motivation, and how your success will be judged. Most projects will not have this definition prior to commencement, so this is what the first phase of the design process is about.

#### ii) Defining the Work Processes

Although ostensibly involved in the storage and retrieval of data, the majority of database systems support one or more work processes. The users are not just storing data, they need to use it in their applications. Understanding the work processes the data needs to support is crucial to understanding the semantics of the data model.

#### iii) Building the Conceptual Data Model

More than simply a set of table structures, the conceptual data model defines the data usage for the entire system. This includes not only the logical data model, but also a description of how the work processes interact with the data.

#### iv) Preparing the Database Schema

The database schema translates the conceptual data model into physical terms. It includes a description of the tables that will be implemented in the system and also the physical architecture of the data.

#### v) Designing the User Interface

No matter how impressive the technical performance of the system, if the user interface is not easy-to-use, confusing, or patronizing, the project is unlikely to be successful. To most users, after all, the interface is the system. In human-computer interaction and computer science, there exists the concept of usability, that usually refers to the elegance and clarity with which the interaction with a computer program or a web site is designed. In database design it is important to implement this concept of usability.

## 2.2 RELATIONAL DATABASE DESIGN

### 2.2.1 Features of good relational designs

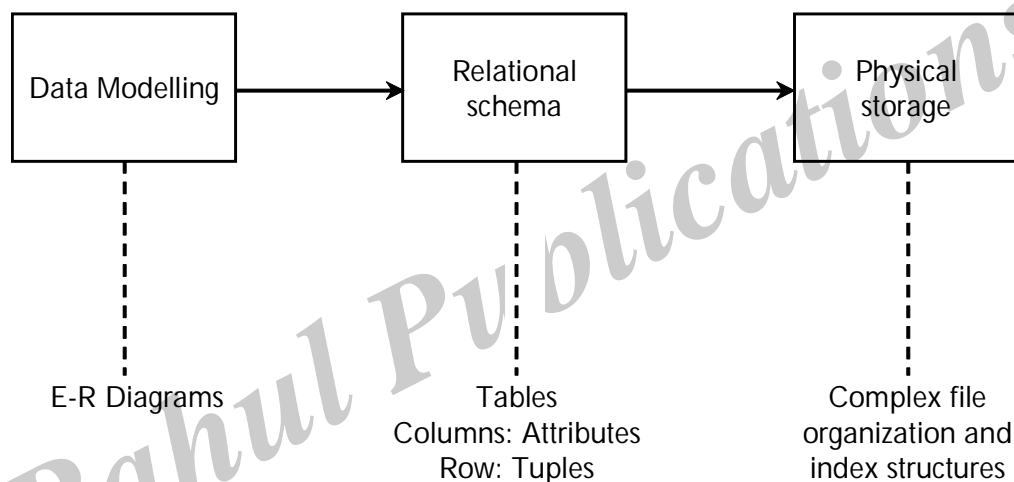
#### Q27. What is relational database design ?

*Ans :*

(Imp.)

The relational data model was introduced by C. F. Codd in 1970. Currently, it is the most widely used data model. The relational data model describes the world as “a collection of inter-related relations (or tables).” A relational data model involves the use of data tables that collect groups of elements into relations. These models work based on the idea that each table setup will include a primary key or identifier. Other tables use that identifier to provide “relational” data links and results.

Today, there are many commercial Relational Database Management System (RDBMS), such as Oracle, IBM DB2, and Microsoft SQL Server. There are also many free and open-source RDBMS, such as MySQL, mSQL (mini-SQL) and the embedded Java DB (Apache Derby). Database administrators use Structured Query Language (SQL) to retrieve data elements from a relational database.



As mentioned, the primary key is a fundamental tool in creating and using relational data models. It must be unique for each member of a data set. It must be populated for all members. Inconsistencies can cause problems in how developers retrieve data. Other issues with relational database designs include excessive duplication of data, faulty or partial data, or improper links or associations between tables. A large part of routine database administration involves evaluating all the data sets in a database to make sure that they are consistently populated and will respond well to SQL or any other data retrieval method.

For example, a conventional database row would represent a tuple, which is a set of data that revolves around an instance or virtual object so that the primary key is its unique identifier. A column name in a data table is associated with an attribute, an identifier or feature that all parts of a data set have. These and other strict conventions help to provide database administrators and designers with standards for crafting relational database setups.

#### Q28. What are the major objectives of good database design.

*Ans :*

(Imp.)

- **Eliminate Data Redundancy:** the same piece of data shall not be stored in more than one place. This is because duplicate data not only waste storage spaces but also easily lead to inconsistencies.

- **Ensure Data Integrity and Accuracy:** is the maintenance of, and the assurance of the accuracy and consistency of, data over its entire life-cycle, and is a critical aspect to the design, implementation, and usage of any system which stores, processes, or retrieves data.

**The relational model has provided the basis for:**

- Research on the theory of data/relationship/constraint
- Numerous database design methodologies
- The standard database access language called structured query language (SQL)
- Almost all modern commercial database management systems

Relational databases go together with the development of SQL. The simplicity of SQL - where even a novice can learn to perform basic queries in a short period of time - is a large part of the reason for the popularity of the relational model.

The two tables below relate to each other through the product code field. Any two tables can relate to each other simply by creating a field they have in common.

**Table 1**

Product_code	Description	Price
A416	Colour Pen	25.00
C923	Pencil box	45.00

**Table 2**

Invoice_code	Invoice_line	Product_code	Quantity
3804	1	A416	15
3804	2	C923	24

There are four stages of an RDM which are as follows:

- **Relations and attributes:** The various tables and attributes related to each table are identified. The tables represent entities, and the attributes represent the properties of the respective entities.
- **Primary keys:** The attribute or set of attributes that help in uniquely identifying a record is identified and assigned as the primary key.
- **Relationships:** The relationships between the various tables are established with the help of foreign keys. Foreign keys are attributes occurring in a table that are primary keys of another table. The types of relationships that can exist between the relations (tables) are One to one, One to many, and Many to many.
- **Normalization:** This is the process of optimizing the database structure. Normalization simplifies the database design to avoid redundancy and confusion. The different normal forms are as follows:
  1. First normal form
  2. Second normal form
  3. Third normal form
  4. Boyce-Codd normal form
  5. Fifth normal form

By applying a set of rules, a table is normalized into the above normal forms in a linearly progressive fashion. The efficiency of the design gets better with each higher degree of normalization.

**Q29. What are the advantages of relational databases.**

*Ans :*

- The main advantages of relational databases are that they enable users to easily categorize and store data that can later be queried and filtered to extract specific information for reports.
- Relational databases are also easy to extend and aren't reliant on the physical organization.
- After the original database creation, a new data category can be added without all existing applications being modified.
- **Accurate:** Data is stored just once, which eliminates data deduplication.
- **Flexible:** Complex queries are easy for users to carry out.
- **Collaborative:** Multiple users can access the same database.
- **Trusted:** Relational database models are mature and well-understood.
- **Secure:** Data in tables within relational database management systems (RDBMS) can be limited to allow access by only particular users.

**2.2.2 Atomic domains and first normal form**

**Q30. What is meant by Normalization ?**

*Ans :*

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes:

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

**Q31. What are the problems normalization with an example.**

*Ans :*

If a table is not properly normalized and have data redundancy then it will not only eat up extra memory space but will also make it difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion Anomalies are very frequent if database is not normalized. To understand these anomalies let us take an example of a Student table.

rollNo	name	branch	hod	office_tel
401	Akon	CSE	Mr. X	53337
402	Bkon	CSE	Mr. X	53337
403	Ckon	CSE	Mr. X	53337
404	Dkon	CSE	Mr. X	53337

In the table above, we have data of 4 Computer Sci. students. As we can see, data for the fields branch, hod (Head of Department) and office\_tel is repeated for the students who are in the same branch in the college, this is Data Redundancy.

**i) Insertion Anomaly**

Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as NULL.

Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students.

These scenarios are nothing but Insertion anomalies.

**ii) Updation Anomaly**

What if Mr. X leaves the college? or is no longer the HOD of computer science department? In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency. This is Updation anomaly.

**iii) Deletion Anomaly**

In our Student table, two different informations are kept together, Student information and Branch information. Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is Deletion anomaly.

**Q32. What is Redundancy ? How it create Problem in Database ?**

*Ans :*

Data redundancy is a condition created within a database or data storage technology in which the same piece of data is held in two separate places.

This can mean two different fields within a single database, or two different spots in multiple software environments or platforms. Whenever data is repeated, this basically constitutes data redundancy. This can occur by accident, but is also done deliberately for backup and recovery purposes.

Redundancy means having multiple copies of same data in the database. This problem arises when a database is well normalized. Suppose a table of student details attributes are: student Id, student name, college name, college rank, course opted.

StudentJD	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	Btech	1
101	Ankit	7900734858	GEU	Btech	1
102	Aysuh	7300936759	GEU	Btech	1
103	Ravi	7300901556	GEU	Btech	1

As it can be observed that values of attribute college name, college rank, course is being repeated which can lead to problems. Problems caused due to redundancy are:

1. Insertion anomaly,
2. Deletion anomaly, and
3. Updation anomaly.

**1. Insertion Anomaly**

If a student detail has to be inserted whose course is not being decided yet then insertion will not be possible till the time course is decided for student.

StudentJD	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	Btech	1

This problem happens when the insertion of a data record is not possible without adding some additional unrelated data to the record.

**2. Deletion Anomaly**

If the details of students in this table is deleted then the details of college will also get deleted which should not occur by common sense.

This anomaly happens when deletion of a data record results in losing some unrelated information that was stored as part of the record that was deleted from a table.



### 3. Updation Anomaly

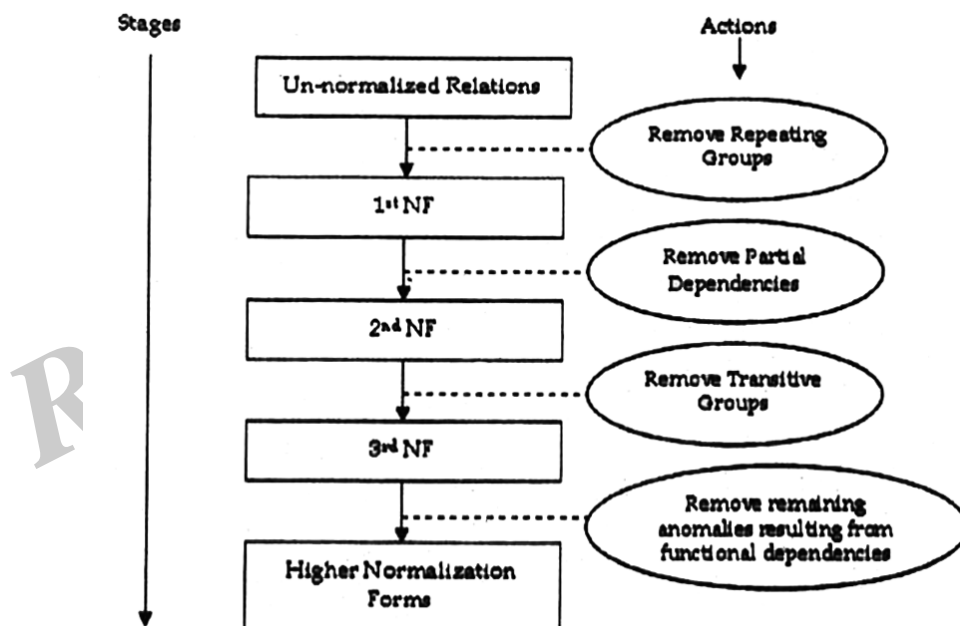
Suppose if the rank of the college changes then changes will have to be all over the database which will be time-consuming and computationally costly.

Student_ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	Btech	1
101	Ankit	7900734858	GEU	Btech	1
102	Aysuh	7300936759	GEU	Btech	1
103	Ravi	7300901556	GEU	Btech	1

All places should be updated

If updation do not occur at all places then database will be in inconsistent state.

### NORMALIZATION PROCESS



### Normalization Rule

Normalization rules are divided into the following normal forms:

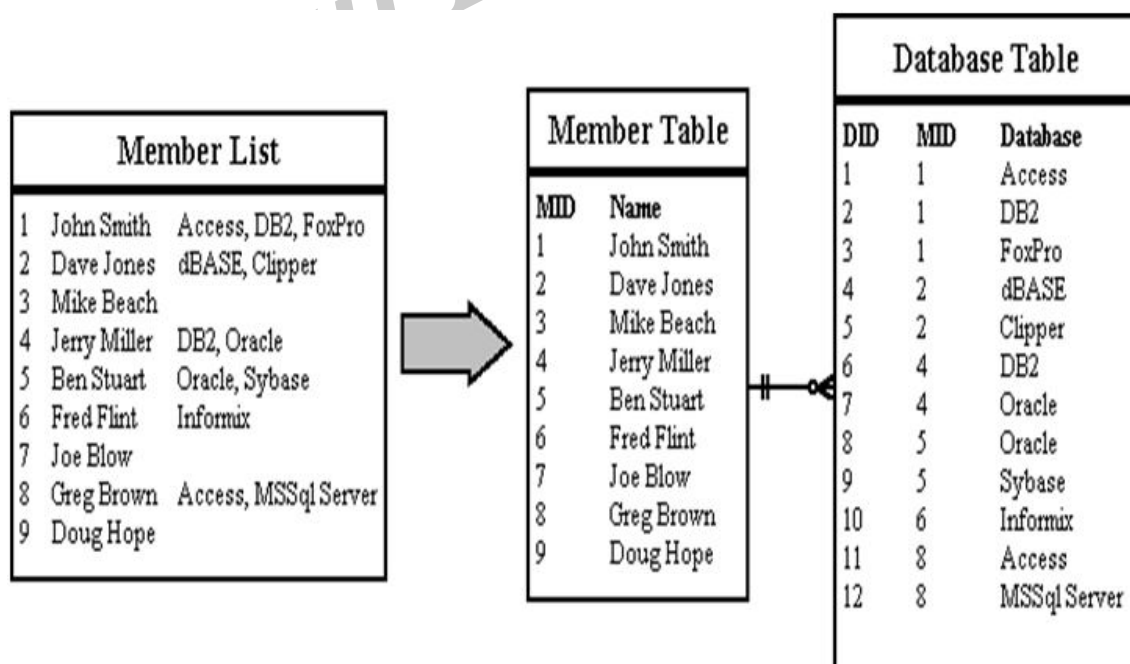
- UN- Normalized form ( UNF )
- First Normal Form (1NF)
- Second Normal Form(2NF)
- Third Normal Form(3NF)
- BCNF
- Fourth Normal Form(4NF)

**Q33. Explain the Rules of Normalization? Following are the 5 Rules of Data Normalization***Ans :*

- i) **Eliminate Repeating Groups:** Make a separate table for each set of related attributes, and give each table a primary key.
  - ii) **Eliminate Redundant Data:** If an attribute depends on only part of a multi valued key, remove it to a separate table.
  - iii) **Eliminate Columns Not Dependent On Key:** If attributes do not contribute to a description of the key, remove them to a separate table.
  - iv) **Isolate Independent Multiple Relation-ships:** No table may contain two or more 1:n or n:m relationships that are not directly related.
  - v) **Isolate Semantically Related Multiple Relationships:** There may be practical constraints on information that justify separating logically related many-to-many relationships.
- i) **Eliminate Repeating Groups**

In the original member list, each member name is followed by any databases that the member has experience with. Some might know many, and others might not know any. To answer the question, "Who knows DB2?" we need to perform an awkward scan of the list looking for references to DB2. This is inefficient and an extremely untidy way to store information.

Moving the known databases into a separate table helps a lot. Separating the repeating groups of databases from the member information results in **first normal form**. The MemberID in the database table matches the primary key in the member table, providing a foreign key for relating the two tables with a join operation. Now we can answer the question by looking in the database table for "DB2" and getting the list of members.



**ii) Eliminate Redundant Data**

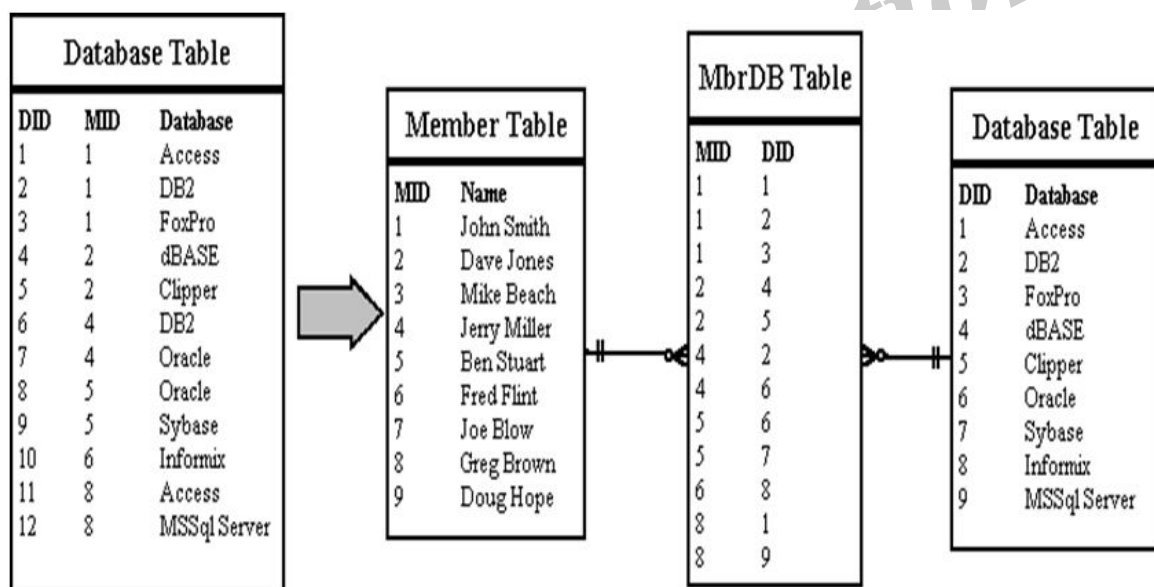
In the Database Table, the primary key is made up of the MemberID and the DatabaseID. This makes sense for the "Where Learned" and "Skill Level" attributes, since they will be different for every member/database combination. But the database name depends only on the DatabaseID. The same database name will appear redundantly every time its associated ID appears in the Database Table.

Suppose you want to reclassify a database - give it a different DatabaseID. The change has to be made for every member that lists that database! If you miss some, you'll have several members with the same database under different IDs. This is an update anomaly.

Or suppose the last member listing a particular database leaves the group. His records will be removed from the system, and the database will not be stored anywhere! This is a delete anomaly. To avoid these problems, we need second normal form.

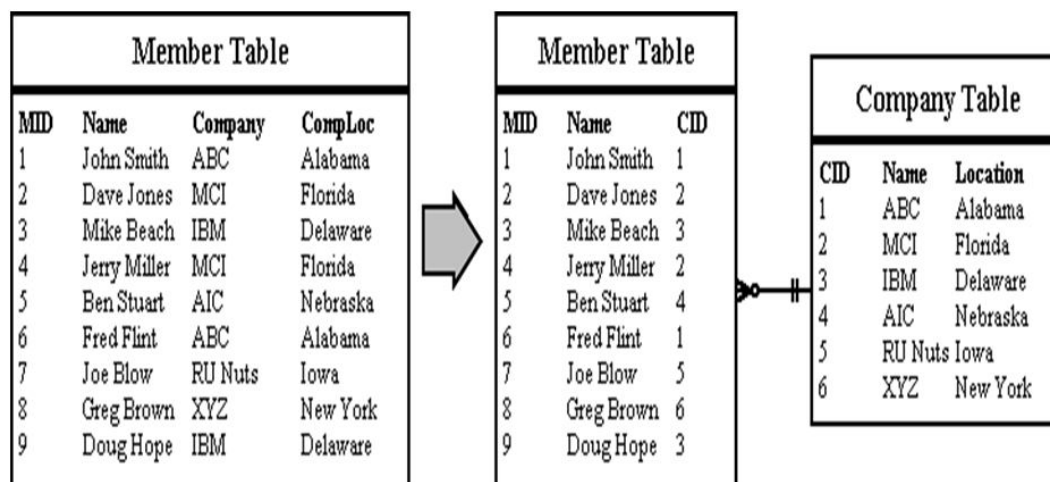
To achieve this, separate the attributes depending on both parts of the key from those depending only on the DatabaseID. This results in two tables: "Database" which gives the name for each DatabaseID, and "MemberDatabase" which lists the databases for each member.

Now we can reclassify a database in a single operation: look up the DatabaseID in the "Database" table and change its name. The result will instantly be available throughout the application.

**iii) Eliminate Columns Not Dependent On Key**

The Member table satisfies first normal form - it contains no repeating groups. It satisfies second normal form - since it doesn't have a multivalued key. But the key is MemberID, and the company name and location describe only a company, not a member. To achieve third normal form, they must be moved into a separate table. Since they describe a company, CompanyCode becomes the key of the new "Company" table.

The motivation for this is the same for second normal form: we want to avoid update and delete anomalies. For example, suppose no members from the IBM were currently stored in the database. With the previous design, there would be no record of its existence, even though 20 past members were from IBM!



#### iv) Isolate Independent Multiple Relationships

This applies only to designs that include one-to-many and many-to-many relationships. An example of one-to-many is that one company can employ many members. An example of a many-to-many relationship is that a member can know many databases and several members might know the same database.

Suppose we want to add a new attribute to the Member Database table called "Attire". This way we can look for members that not only know DB2, but also typically wear a suit and tie. **Fourth normal** form dictates against this (using the Member Database table, not wearing suits and ties). The two attributes do not share a meaningful relationship. A member may know a certain database, and he/she might just wear a wet suit. This doesn't mean he/she can do both at the same time (unless you have a water-proof computer terminal). How will you represent this if you store both attributes in the same table.

#### v) Isolate Semantically Related Multiple Relationships

Usually, related attributes belong together. For example, if we really wanted to record which databases each member works on wearing which kinds of clothes, we would want to keep the attire attribute in the MemberDatabase table. But there are times when special characteristics of the data make it more efficient to separate even logically related attributes.

Imagine that our system will record which jobs are available in each company, and which schools typically supply candidates to those companies. This suggests a CompanySchoolJob table which satisfies **fourth normal form**. As long as any company can use any candidates from any school, this works fine.

Now suppose a law is passed to prevent exclusive arrangements: a company accepting candidates must accept them from all schools it deals with. In other words, if IBM is hiring DBAs and wants to maintain a relationship with MIT, it must accept DBAs from MIT.

The need for **fifth normal form** becomes clear when we consider inserts and deletes. Suppose a company decides to create 3 new jobs types: HTML DBA, Java Programmer and Underwater DB2 DBA. Suppose further that it already deals with three schools that can supply candidates for those positions. This will require nine new rows in the database, one for each school/job combination.

Breaking up the table reduces the number of inserts to six. Here are the tables necessary for **fifth normal form**, shown with the six newly inserted rows in bold type. If an application involves significant update activity, fifth normal form can mean important savings. Note that these combination tables develop naturally out of entity-relationship analysis.

**Q34. Explain first normal form (1NF) in detail.***Ans .:***(Imp.)**

He first normal form expects you to follow a few simple rules while designing your database, and they are:

**Rule 1: Single Valued Attributes**

Each column of your table should be single valued which means they should not contain multiple values. We will explain this with help of an example later, let's see the other rules for now.

**Rule 2: Attribute Domain should not change**

This is more of a "Common Sense" rule. In each column the values stored must be of the same kind or type.

**For example:** If you have a column dob to save date of births of a set of people, then you cannot or you must not save 'names' of some of them in that column along with 'date of birth' of others in that column. It should hold only 'date of birth' for all the records/rows.

**Rule 3: Unique name for Attributes/Columns**

This rule expects that each column in a table should have a unique name. This is to avoid confusion at the time of retrieving data or performing any other operation on the stored data.

If one or more columns have same name, then the DBMS system will be left confused.

**Rule 4: Order doesn't matters**

This rule says that the order in which you store the data in your table doesn't matter.

**Example**

Although all the rules are self explanatory still let's take an example where we will create a table to store student data which will have student's roll no., their name and the name of subjects they have opted for.

Here is our table, with some sample data added to it.

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

Our table already satisfies 3 rules out of the 4 rules, as all our column names are unique, we have stored data in the order we wanted to and we have not inter-mixed different type of data in columns.

But out of the 3 different students in our table, 2 have opted for more than 1 subject. And we have stored the subject names in a single column. But as per the 1st Normal form each column must contain atomic value.

**To solve this problem** It's very simple, because all we have to do is break the values into atomic values.

Here is our updated table and it now satisfies the First Normal Form.

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C + +

By doing so, although a few values are getting repeated but values for the subject column are now atomic for each record/row.

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

### 2.2.3 Decomposition using functional dependencies, functional dependency theory, Decomposition using multi valued dependencies

#### Q35. What is Functional Dependency ?

*Ans :*

Functional Dependency is when one attribute determines another attribute in a DBMS system. Functional Dependency plays a vital role to find the difference between good and bad database design.

#### Example

Employee number	Employee Name	Salary	City
1.	Dana	50000	San Francisco
2.	Francis	38000	London
3.	Andrew	25000	Tokyo

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc.

By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

A functional dependency is denoted by an arrow  $\rightarrow$

The functional dependency of X on Y is represented by  $X \rightarrow Y$ .

#### Q36. List and explain types of Functional Dependencies ?

*Ans :*

(Imp.)

1. Multivalued dependency
2. Trivial functional dependency
3. Non-trivial functional dependency
4. Transitive dependency

### 1. Multivalued dependency in DBMS

Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table. A multivalued dependency is a complete constraint between two sets of attributes in a relation. It requires that certain tuples be present in a relation.

#### Example

Car_model	Maf_year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue
H010	2015	Metallic
H033	2012	Gray

In this example, maf\_year and color are independent of each other but dependent on car\_model. In this example, these two columns are said to be multivalued dependent on car\_model.

This dependence can be represented like this:

car\_model  $\twoheadrightarrow$  maf\_year

car\_model  $\twoheadrightarrow$  colour

### 2. Trivial Functional dependency

The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.

So,  $X \rightarrow Y$  is a trivial functional dependency if  $Y$  is a subset of  $X$ .

#### For example

Emp_id	Emp_name
AS555	Harry
AS811	George
AS999	Kevin

Consider this table with two columns Emp\_id and Emp\_name.

$\{Emp\_id, Emp\_name\} \rightarrow Emp\_id$  is a trivial functional dependency as Emp\_id is a subset of  $\{Emp\_id, Emp\_name\}$ .

### 3. Non trivial functional dependency in DBMS

Functional dependency which also known as a nontrivial dependency occurs when  $A \rightarrow B$  holds true where  $B$  is not a subset of  $A$ . In a relationship, if attribute  $B$  is not a subset of attribute  $A$ , then it is considered as a non-trivial dependency.

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Apple	Tim Cook	57

#### Example

$\{Company\} \rightarrow \{CEO\}$  (if we know the Company, we know the CEO name)

But CEO is not a subset of Company, and hence it's non-trivial functional dependency.

### 4. Transitive dependency

A transitive is a type of functional dependency which happens when to is indirectly formed by two functional dependencies.

#### Example

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

$\{Company\} \rightarrow \{CEO\}$  (if we know the company, we know its CEO's name)

$\{CEO\} \rightarrow \{Age\}$  If we know the CEO, we know the Age

Therefore according to the rule of transitive dependency:

$\{Company\} \rightarrow \{Age\}$  should hold, that makes sense because if we know the company name, we can know his age.

#### Advantages

- Functional Dependency avoids data redundancy. Therefore same data do not repeat at multiple locations in that database
- It helps you to maintain the quality of data in the database

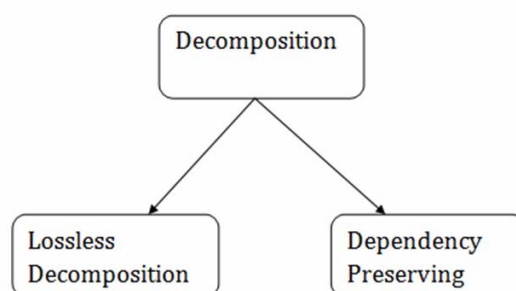
- It helps you to defined meanings and constraints of databases
- It helps you to identify bad designs
- It helps you to find the facts regarding the database design

**Q37. Define Decomposition ? What is the use of Decomposition ? Explain its types ?**

*Ans :*

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

**Types of Decomposition**



**i) Lossless Decomposition**

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

**Example**

**EMPLOYEE\_DEPARTMENT Table**

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing



The above relation is decomposed into two relations EMPLOYEE and DEPARTMENT

**EMPLOYEE table:**

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY
22	Denim	28	Mumbai
33	Alina	25	Delhi
46	Stephan	30	Bangalore
52	Katherine	36	Mumbai
60	Jack	40	Noida

**DEPARTMENT table**

DEPT_ID	EMP_ID	DEPT_NAME
827	22	Sales
438	33	Marketing
869	46	Finance
575	52	Production
678	60	Testing

Now, when these two relations are joined on the common column "EMP\_ID", then the resultant relation will look like:

**Employee ⋈ Department**

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

Hence, the decomposition is Lossless join decomposition.

## ii) Dependency Preservation

- It is an important constraint of the database.
- In the dependency preservation, at least one decomposed table must satisfy every dependency.
- If a relation R is decomposed into relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or must be derivable from the combination of functional dependencies of R1 and R2.
- For example, suppose there is a relation R (A, B, C, D) with functional dependency set (A → BC). The relational R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A → BC is a part of relation R1(ABC).

## Attribute preservation

This is simple requirement that involves preserving all the attributes that were there in the relation that is being decomposed.

All attributes must be preserved through the process of normalization.

Start with universal relation schema R

$R = \{A_1, A_2, A_3, \dots, A_n\}$ , the set of attributes

D is a decomposition of R such that

$D = \{R_1, R_2, \dots, R_m\}$  and  $R = \bigcup R_i$ .

## 2.2.4 Normal forms - 2NF, 3NF and BCNF

### Q38. Explain in detail second Normal Form (2NF)

*Ans :*

#### Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

#### Example

Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

**Candidate Keys:** {teacher\_id, subject}

**Non prime attribute:** teacher\_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher\_age is dependent on teacher\_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says “**no** non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this:

**teacher\_details table:**

teacher_id	teacher_age
111	38
222	38
333	40

teacher\_subject table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

**Q39. Explain in detail Third Normal Form (3NF).**

*Ans :*

(Imp.)

### Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency  $X \rightarrow Y$  at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

### Example

Suppose a company wants to store the complete address of each employee, they create a table named employee\_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

**Super keys:** {emp\_id}, {emp\_id, emp\_name}, {emp\_id, emp\_name, emp\_zip}...so on

**Candidate Keys:** {emp\_id}

**Non-prime attributes:** all attributes except emp\_id are non-prime as they are not part of any candidate keys.

Here, emp\_state, emp\_city&emp\_district dependent on emp\_zip. And, emp\_zip is dependent on emp\_id that makes non-prime attributes (emp\_state, emp\_city&emp\_district) transitively dependent on super key (emp\_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

**employee table**

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

**employee\_zip table**

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

**Q40. Explain in detail Boycecodd Normal Form (BCNF).**

*Ans :*

(Imp.)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency  $X \rightarrow Y$ , X should be the super key of the table.

**Example**

Suppose there is a company wherein employees work in more than one department. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

**Functional dependencies in the table above**

emp\_id  $\rightarrow$  emp\_nationality

emp\_dept  $\rightarrow$  {dept\_type, dept\_no\_of\_emp}

**Candidate key:** {emp\_id, emp\_dept}

The table is not in BCNF as neither emp\_id nor emp\_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:

**emp\_nationality table**

emp_id	emp_nationality
1001	Austrian
1002	American

**emp\_dept table**

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

**emp\_dept\_mapping table**

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

**Functional dependencies**

emp\_id -> emp\_nationality

emp\_dept -> {dept\_type, dept\_no\_of\_emp}

**Candidate keys**

For first table: emp\_id

For second table: emp\_dept

For third table: {emp\_id, emp\_dept}

This is now in BCNF as in both the functional dependencies left side part is a key

**Q41. List out the advantages and disadvantages of Normalization.**

*Ans :*

**Advantages of Normalization**

Here we can see why normalization is an attractive prospect in RDBMS concepts.

1. A smaller database can be maintained as normalization eliminates the duplicate data. Overall size of the database is reduced as a result.
2. Better performance is ensured which can be linked to the above point. As databases become lesser in size, the passes through the data becomes faster and shorter thereby improving response time and speed.
3. Narrower tables are possible as normalized tables will be fine-tuned and will have lesser columns which allows for more data records per page.

4. Fewer indexes per table ensures faster maintenance tasks (index rebuilds).
5. Also realizes the option of joining only the tables that are needed.

### Disadvantages of Normalization

1. More tables to join as by spreading out data into more tables, the need to join table's increases and the task becomes more tedious. The database becomes harder to realize as well.
2. Tables will contain codes rather than real data as the repeated data will be stored as lines of codes rather than the true data. Therefore, there is always a need to go to the lookup table.
3. Data model becomes extremely difficult to query against as the data model is optimized for applications, not for ad hoc querying. (Ad hoc query is a query that cannot be determined before the issuance of the query. It consists of an SQL that is constructed dynamically and is usually constructed by desktop friendly query tools.). Hence it is hard to model the database without knowing what the customer desires.
4. As the normal form type progresses, the performance becomes slower and slower.
5. Proper knowledge is required on the various normal forms to execute the normalization process efficiently. Careless use may lead to terrible design filled with major anomalies and data inconsistency.

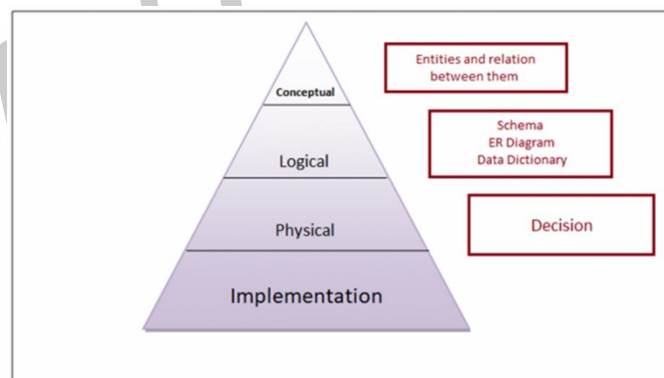
### 2.2.5 The database design methodology for Relational databases.

#### Q42. Discuss database design methodologies

*Ans :*

Database Design Methodologies has phases to guide the designer for assistance. The Methodology has a structured approach to help in the design process.

The following are the phases/ models:



#### Conceptual Phase

The Conceptual phase lets you know the entities and the relation between them. It describes the conceptual schema. The entities & relations are defined here.

#### Logical Phase

Logical data model provides details about the data to the physical phase. The physical process gives ER Diagram, data dictionary, schema, etc that acts as a source for the physical design process.

#### Physical Phase

The physical database design allows the designer to decide on how the database will be implemented.

## Short Question and Answers

### 1. What is database design?

*Ans :*

#### Meaning

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

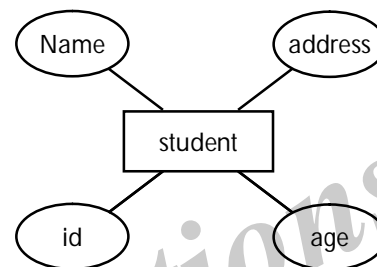
### 2. What is ER Model ?

*Ans :*

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

#### For example

Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



### 3. What is Data Redundancy?

*Ans :*

Data redundancy occurs in database systems which have a field that is repeated in two or more tables. When customer data is duplicated and attached with each product bought, then redundancy of data is a known source of inconsistency, since the entity "customer" might appear with different values for a given attribute.

Data redundancy leads to data anomalies and corruption and should be avoided when creating a relational database consisting of several entities. Database normalization prevents redundancy and makes the best possible usage of storage. The proper use of foreign keys can minimize data redundancy and reduce the chance of destructive anomalies appearing. Concerns with respect to the efficiency and convenience can sometimes result in redundant data design despite the risk of corrupting the data.

### 4. How does data redundancy occur?

*Ans :*

Data redundancy can be designed; for example, suppose you want to back up your company's data nightly. This creates a redundancy. Data redundancy can also occur by mistake. For

example, the database designer who created a system with a new record for each sale may not have realized that his design caused the same address to be entered repeatedly. You may also end up with redundant data when you store the same information in multiple systems. For instance, suppose you store the same basic employee information in Human Resources records and in records maintained for your local site office.

## 5. List out the benefits of ER Diagrams

*Ans :*

ER diagrams constitute a very useful framework for creating and manipulating databases.

- First, ER diagrams are easy to understand and do not require a person to undergo extensive training to be able to work with it efficiently and accurately.
- This means that designers can use ER diagrams to easily communicate with developers, customers, and end users, regardless of their IT proficiency.
- Second, ER diagrams are readily translatable into relational tables which can be used to quickly build databases. In addition, ER diagrams can directly be used by database developers as the blueprint for implementing data in specific software applications.
- Lastly, ER diagrams may be applied in other contexts such as describing the different relationships and operations within an organization.

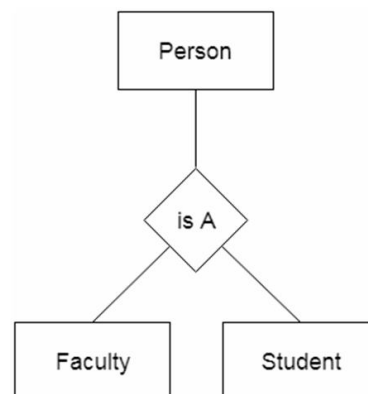
## 6. Explain the concept of Generalization.

*Ans :*

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.

- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

**For example,** Faculty and Student entities can be generalized and create a higher level entity Person.



## 7. Specialization

*Ans :*

- Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

## 8. Aggregation

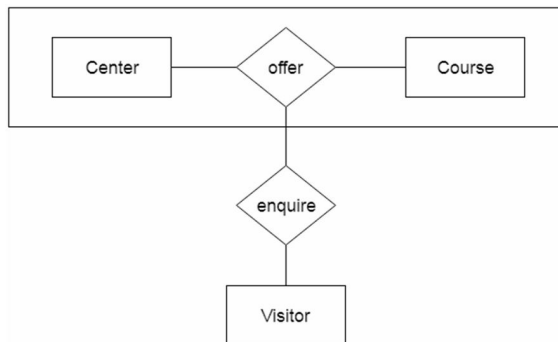
*Ans :*

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

### For example

Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.





## 9. Normalization

*Ans :*

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes:

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

## 10. Rules Data Normalization.

*Ans :*

- i) **Eliminate Repeating Groups:** Make a separate table for each set of related attributes, and give each table a primary key.
- ii) **Eliminate Redundant Data:** If an attribute depends on only part of a multi valued key, remove it to a separate table.
- iii) **Eliminate Columns Not Dependent On Key:** If attributes do not contribute to a description of the key, remove them to a separate table.
- iv) **Isolate Independent Multiple Relationships:** No table may contain two or more 1:n or n:m relationships that are not directly related.

- v) **Isolate Semantically Related Multiple Relationships:** There may be practical constraints on information that justify separating logically related many-to-many relationships.

## 11. Explain first normal form (1NF) in detail.

*Ans :*

The first normal form expects you to follow a few simple rules while designing your database, and they are:

### Rule 1: Single Valued Attributes

Each column of your table should be single valued which means they should not contain multiple values. We will explain this with help of an example later, let's see the other rules for now.

### Rule 2: Attribute Domain should not change

This is more of a "Common Sense" rule. In each column the values stored must be of the same kind or type.

**For example:** If you have a column 'dob' to save date of births of a set of people, then you cannot or you must not save 'names' of some of them in that column along with 'date of birth' of others in that column. It should hold only 'date of birth' for all the records/rows.

### Rule 3: Unique name for Attributes/Columns

This rule expects that each column in a table should have a unique name. This is to avoid confusion at the time of retrieving data or performing any other operation on the stored data.

If one or more columns have same name, then the DBMS system will be left confused.

### Rule 4: Order doesn't matters

This rule says that the order in which you store the data in your table doesn't matter.

## 12. What is Functional Dependency ?

*Ans :*

Functional Dependency is when one attribute determines another attribute in a DBMS system. Functional Dependency plays a vital role to find the difference between good and bad database design.

**Example**

Employee number	Employee Name	Salary	City
1.	Dana	50000	San Francisco
2.	Francis	38000	London
3.	Andrew	25000	Tokyo

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc.

By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

A functional dependency is denoted by an arrow  $\rightarrow$

The functional dependency of X on Y is represented by  $X \rightarrow Y$ .

**13. Define Decomposition ?**

*Ans :*

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

**14. Explain in detail second Normal Form (2NF)**

*Ans :*

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

**Example**

Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

**Candidate Keys:** {teacher\_id, subject}

**Non prime attribute:** teacher\_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher\_age is dependent on teacher\_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says “**no** non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this:

**teacher\_details table:**

teacher_id	teacher_age
111	38
222	38
333	40

**teacher\_subject table:**

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

### 15. Explain in detail Third Normal Form (3NF).

*Ans :*

#### Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency  $X \rightarrow Y$  at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

#### Example

Suppose a company wants to store the complete address of each employee, they create a table named employee\_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

**Super keys:** {emp\_id}, {emp\_id, emp\_name}, {emp\_id, emp\_name, emp\_zip}...so on

**Candidate Keys:** {emp\_id}

**Non-prime attributes:** all attributes except emp\_id are non-prime as they are not part of any candidate keys.

Here, emp\_state, emp\_city&emp\_district dependent on emp\_zip. And, emp\_zip is dependent on emp\_id that makes non-prime attributes (emp\_state, emp\_city&emp\_district) transitively dependent on super key (emp\_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

**employee table**

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

**employee\_zip table**

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

#### 16. Explain in detail Boyceodd Normal Form (BCNF).

*Ans :*

(Imp.)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency  $X \rightarrow Y$ , X should be the super key of the table.

#### Example

Suppose there is a company wherein employees work in more than one department. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

**Functional dependencies in the table above**

emp\_id -> emp\_nationality

emp\_dept -> {dept\_type, dept\_no\_of\_emp}

**Candidate key:** {emp\_id, emp\_dept}

The table is not in BCNF as neither emp\_id nor emp\_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:

**emp\_nationality table**

emp_id	emp_nationality
1001	Austrian
1002	American

**emp\_dept table**

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

**emp\_dept\_mapping table**

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

**Functional dependencies**

emp\_id -> emp\_nationality

emp\_dept -> {dept\_type, dept\_no\_of\_emp}

**Candidate keys**

For first table: emp\_id

For second table: emp\_dept

For third table: {emp\_id, emp\_dept}

This is now in BCNF as in both the functional dependencies left side part is a key

**17. Advantages of Normalization***Ans :*

Here we can see why normalization is an attractive prospect in RDBMS concepts.

1. A smaller database can be maintained as normalization eliminates the duplicate data. Overall size of the database is reduced as a result.
  2. Better performance is ensured which can be linked to the above point. As databases become lesser in size, the passes through the data becomes faster and shorter thereby improving response time and speed.
  3. Narrower tables are possible as normalized tables will be fine-tuned and will have lesser columns which allows for more data records per page.
  4. Fewer indexes per table ensures faster maintenance tasks (index rebuilds).
  5. Also realizes the option of joining only the tables that are needed.
- 

**18. Disadvantages of Normalization***Ans :*

1. More tables to join as by spreading out data into more tables, the need to join table's increases and the task becomes more tedious. The database becomes harder to realize as well.
2. Tables will contain codes rather than real data as the repeated data will be stored as lines of codes rather than the true data. Therefore, there is always a need to go to the lookup table.
3. Data model becomes extremely difficult to query against as the data model is optimized for applications, not for ad hoc querying. (Ad hoc query is a query that cannot be determined before the issuance of the query. It consists of an SQL that is constructed dynamically and is usually constructed by desktop friendly query tools.). Hence it is hard to model the database without knowing what the customer desires.
4. As the normal form type progresses, the performance becomes slower and slower.
5. Proper knowledge is required on the various normal forms to execute the normalization process efficiently. Careless use may lead to terrible design filled with major anomalies and data inconsistency.

### Choose the Correct Answers

1. In SQL which command is used to change a table structure [ a ]
  - a) Alter table
  - b) Modify table
  - c) Change table
  - d) All
2. A command that lets you change one or more fields in a records is [ b ]
  - a) insert
  - b) modify
  - c) look-up
  - d) all
3. Which SQL constraints is used to retrieve only unique value ? [ b ]
  - a) Distinctive
  - b) Unique
  - c) Distinct
  - d) Different
4. Which SQL command is used to retrieve data [ c ]
  - a) delete
  - b) insert
  - c) select
  - d) join
5. The command used to create a database [ a ]
  - a) create database
  - b) make user
  - c) create user
  - d) all
6. \_\_\_\_\_ operator is used for appending two strings [ c ]
  - a) &
  - b) %
  - c) ||
  - d) ~
7. Which is a simple or compound symbol that has special meaning in PL/SQL [ a ]
  - a) delimiters
  - b) identifiers
  - c) literals
  - d) comments
8. Which statement lets us to create stand alone functions [ b ]
  - a) SQL create procedure
  - b) SQL create function
  - c) a & b
  - d) None
9. Which of the following is used to define code that fire an event. [ c ]
  - a) replace
  - b) keywords
  - c) trigger
  - d) cursor
10. Which of the following are implicit cursor attributes. [ b ]
  - a) % found
  - b) % not found
  - c) % row count
  - d) % row type

## *Fill in the blanks*

1. SQL is a language for manipulating databases developed by \_\_\_\_\_
2. SQL commands are used to interact with \_\_\_\_\_
3. All SQL statements start with \_\_\_\_\_
4. \_\_\_\_\_ is also known as nested query.
5. The \_\_\_\_\_ & \_\_\_\_\_ operators are used with where or having clause.
6. The \_\_\_\_\_ clause will evaluate true/false.
7. \_\_\_\_\_ operator is used to negate a condition.
8. Maximum length of characters datatype is \_\_\_\_\_
9. Uniquely identifies each row or column in a database table is \_\_\_\_\_
10. \_\_\_\_\_ is a keyword used to create a table or database.
11. The table alterations are done using \_\_\_\_\_ command
12. \_\_\_\_\_ are used in dataware houses to increase the speed of queries.
13. PL/SQL is called as \_\_\_\_\_ extension of SQL.
14. A \_\_\_\_\_ is a symbol with special meaning.
15. The result of comparison operator is \_\_\_\_\_ or \_\_\_\_\_.

### **ANSWERS**

1. IBM
2. Databases
3. Keyword
4. Subquery
5. Any, All
6. Exists
7. Not
8. 8,000
9. Primary key
10. Create
11. Alter
12. Materialised views
13. Procedural language
14. Delimiter
15. True/False



## UNIT III

**Introduction to SQL:** Overview of the SQL Query Language, SQL Data Definition, Basic Structure of SQL Queries, Additional Basic Operations, Set Operations, Null Values, Aggregate Functions, Nested Subqueries, Modification of the Database.

**Intermediate SQL:** Join Expressions, Views, Transactions, Integrity Constraints, SQL Data Types and Schemas, Authorization.

**Advanced SQL:** Accessing SQL from a Programming Language, Functions and Procedures, Triggers, Recursive Queries.

### 3.1 INTRODUCTION TO SQL

#### 3.1.1 Over Views of SQL

##### Q1. What is SQL ?

*Ans :*

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as:

- MS SQL Server using T-SQL,
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc.

IBM developed the original version of SQL, originally called Sequel, as part of the System R project in the early 1970s. The Sequel language has evolved since then, and its name has changed to SQL (Structured Query Language). Many products now support the SQL language. SQL has clearly established itself as the standard relational database language.

In 1986, the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) published an SQL standard, called

SQL-86. ANSI published an extended standard for SQL, SQL-89, in 1989. The next

version of the standard was SQL-92 standard, followed by SQL:1999, SQL:2003, SQL:2006, and most recently SQL:2008. The bibliographic notes provide references to these standards.

The SQL language has several parts:

##### **Data definition language (DDL)**

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

##### **Data manipulation language (DML)**

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

##### **Integrity**

The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints are disallowed.

##### **View definition**

The SQL DDL includes commands for defining views.

##### **Transaction control**

SQL includes commands for specifying the beginning and ending of transactions.

##### **Embedded SQL and dynamic SQL**

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

##### **Authorization**

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

**Q2. What are the advantages of SQL ?***Ans :*

SQL is widely popular because it offers the following advantages:

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

**Q3. Explain briefly about SQL Process.***Ans :*

When we are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

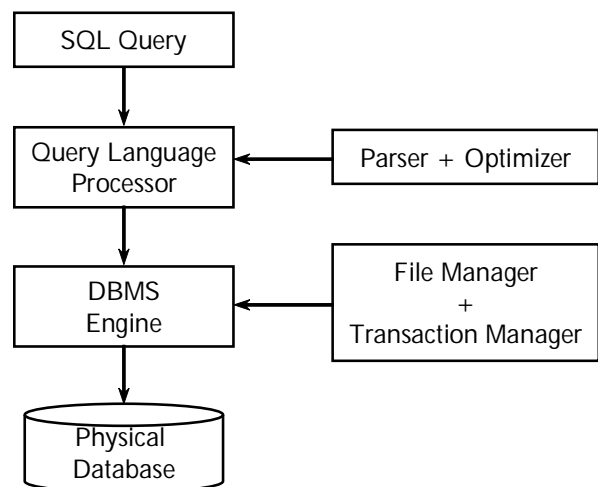
There are various components included in this process.

These components are:

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

Following is a simple diagram showing the SQL Architecture:

**3.1.2 SQL Data definition****Q4. List and explain basic data types supported by SQL ?***Ans :*

The data type of a column defines what value the column can hold: integer, character, money, date and time, binary, and so on.

**SQL Data Types**

Each column in a database table is required to have a name and a data type.

An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

**MySQL Data Types**

In MySQL there are three main data types:

1. Text data types
2. Number data types
3. Date data types

## 1. Text data types

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. <b>Note:</b> If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. <b>Note:</b> The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

## 2. Number data types

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string, allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

\*The integer types have an extra option called UNSIGNED. Normally, the integer goes from an negative to positive value. Adding the UNSIGNED attribute will move that range up so it starts at zero instead of a negative number.

### 3. Date data types

Data type	Description
DATE()	A date. Format: YYYY-MM-DD <b>Note:</b> The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS <b>Note:</b> The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS <b>Note:</b> The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS <b>Note:</b> The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. <b>Note:</b> Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

\*Even if DATETIME and TIMESTAMP return the same format, they work very differently. In an INSERT or UPDATE query, the TIMESTAMP automatically set itself to the current date and time. TIMESTAMP also accepts various formats, like YYYYMMDDHHMISS, YYMMDDHHMISS, YYYYMMDD, or YYMMDD.

### Q5. What is an operator in SQL ? Explain the various types of an SQL ?

*Ans :*

(Imp.)

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

1. Arithmetic operators
2. Comparison operators
3. Logical operators
4. Operators used to negate conditions

#### 1. Arithmetic Operators

Assume 'variable a' holds 10 and 'variable b' holds 20, then

Operator	Description	Example
+	Adds values on either side of the operator.	a + b will give 30
-	Subtracts right hand operand from left hand operand.	a - b will give -10
*	Multiplies values on either side of the operator.	a * b will give 200
/	Divides left hand operand by right hand operand.	b / a will give 2
%	Divides left hand operand by right hand operand and returns remainder.	b % a will give 0

## 2. Comparison Operators

Assume 'variable a' holds 10 and 'variable b' holds 20, then

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a != b) is true.
< >	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(a < > b) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
> =	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a > = b) is not true.
< =	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a < = b) is true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.	(a !< b) is false.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.	(a !> b) is true.

### SQL Logical Operators

Here is a list of all the logical operators available in SQL.

Sr. No.	Operator	Description
1	<b>ALL</b>	The ALL operator is used to compare a value to all values in another value set.
2	<b>AND</b>	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
3	<b>ANY</b>	The ANY operator is used to compare a value to any applicable value in the list as per the condition.
4	<b>BETWEEN</b>	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
5	<b>EXISTS</b>	The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion.
6	<b>IN</b>	The IN operator is used to compare a value to a list of literal values that have been specified.
7	<b>LIKE</b>	The LIKE operator is used to compare a value to similar values using wildcard operators.
8	<b>NOT</b>	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.
9	<b>OR</b>	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
10	<b>IS NULL</b>	The NULL operator is used to compare a value with a NULL value.
11	<b>UNIQUE</b>	The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

#### Q6. List out various types of SQL Commands.

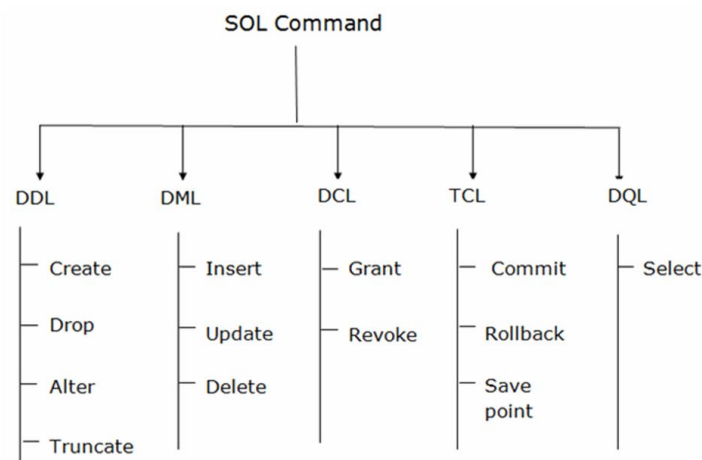
*Ans :*

(Imp.)

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

#### Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



### 1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- (a) CREATE
- (b) ALTER
- (c) DROP
- (d) TRUNCATE

- (a) **CREATE:** It is used to create a new table in the database.

#### Syntax

1. CREATE TABLE TABLE\_NAME (COLUMN\_NAME DATATYPES[,...]);

#### Example

1. CREATE TABLE EMPLOYEE (Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

- (b) **DROP:** It is used to delete both the structure and record stored in the table.

#### Syntax

1. DROP TABLE table\_name;

#### Example

1. DROP TABLE EMPLOYEE;

- (c) **ALTER:** It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

#### Syntax

To add a new column in the table

1. ALTER TABLE table\_name ADD column\_name COLUMN-definition;

To modify existing column in the table:

1. ALTER TABLE table\_name MODIFY(column\_definitions....);

**Example**

1. ALTER TABLE STU\_DETAILS ADD(ADDRESS VARCHAR2(20));
2. ALTER TABLE STU\_DETAILS MODIFY (NAME VARCHAR2(20));

**(d) TRUNCATE:** It is used to delete all the rows from the table and free the space containing the table.

**Syntax**

1. TRUNCATE TABLE table\_name;

**Example**

1. TRUNCATE TABLE EMPLOYEE;

**2. Data Manipulation Language**

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- (a) INSERT
- (b) UPDATE
- (c) DELETE

**(a) INSERT:** The INSERT statement is a SQL query. It is used to insert data into the row of a table.

**Syntax**

1. INSERT INTO TABLE\_NAME
  2. (col1, col2, col3,.... col N)
  3. VALUES (value1, value2, value3, .... valueN);
- (OR)

1. INSERT INTO TABLE\_NAME
2. VALUES (value1, value2, value3, .... valueN);

**For example**

1. INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");

**(b) UPDATE:** This command is used to update or modify the value of a column in the table.

**Syntax**

1. UPDATE table\_name SET [column\_name1= value1,...column\_nameN = valueN]  
[WHERE CONDITION]

**For example**

1. UPDATE students
2. SET User\_Name = 'Sonoo'
3. WHERE Student\_Id = '3'



- (c) **DELETE:** It is used to remove one or more row from a table.

**Syntax**

1. DELETE FROM table\_name [WHERE condition];

**For example**

1. DELETE FROM javatpoint
2. WHERE Author="Sonoo";

**3. Data Control Language**

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- (a) Grant
- (b) Revoke

- (a) **Grant:** It is used to give user access privileges to a database.

**Example**

1. GRANT SELECT, UPDATE ON MY\_TABLE TO SOME\_USER, ANOTHER\_USER;

- (b) **Revoke:** It is used to take back permissions from the user.

**Example**

1. REVOKE SELECT, UPDATE ON MY\_TABLE FROM USER1, USER2;

**4. Transaction Control Language**

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- (a) COMMIT
- (b) ROLLBACK
- (c) SAVEPOINT

- (a) **Commit:** Commit command is used to save all the transactions to the database.

**Syntax**

1. COMMIT;

**Example**

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. COMMIT;

- (b) **Rollback:** Rollback command is used to undo transactions that have not already been saved to the database.

**Syntax**

1. ROLLBACK;

**Example**

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. ROLLBACK;

- (c) **SAVEPOINT:** It is used to roll the transaction back to a certain point without rolling back the entire transaction.

**Syntax**

1. SAVEPOINT SAVEPOINT\_NAME;

5. **Data Query Language**

DQL is used to fetch the data from the database.

It uses only one command:

- (a) **SELECT:** This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

**Syntax**

1. SELECT expressions
2. FROM TABLES
3. WHERE conditions;

**For example**

1. SELECT emp\_name
2. FROM employee
3. WHERE age > 20;

---

**Q7. Describe in detail about DDL commands in SQL ?**

*Ans :*

Data definition language commands are used to create , Modify and delete the structure of the object in the database. The syntax of the DDL command definitely includes a table keyword after the command name.

All DDL command are given below;

1. Create
2. Describe
3. Alter
4. Rename
5. Truncate and
6. Drop

## 1. CREATE

Creating a basic table involves naming the table and defining its columns and each column's data type.

The SQL **CREATE TABLE** statement is used to create a new table.

### Syntax

The basic syntax of the CREATE TABLE statement is as follows:

```
CREATE TABLE table_name(  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    .....  
    columnN datatype,  
    PRIMARY KEY( one or more columns )  
);
```

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Then in brackets comes the list defining each column in the table and what sort of data type it is. The syntax becomes clearer with the following example.

A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement.

### Example

```
CREATE TABLE Employee  
(  
    Emp id number(5),  
    name varchar(20),  
    salary number(10),  
);
```

## 2. Describe

As the name suggests, DESCRIBE is used to describe something. Since in database we have tables, that's why we use **DESCRIBE** or **DESC** (both are same) command to describe the **structure** of a table.

### Syntax

```
DESCRIBE employee;
```

OR

```
DESC employee;
```

Then for the above we will get sample output as

```
Empid    varchar(5)  
Name     varchar(20)  
Salary   number(10)
```

## 3. ALTER

The SQL **ALTER TABLE** command is used to add, delete or modify columns in an existing table. You should also use the ALTER TABLE command to add and drop various constraints on an existing table.

### Syntax

The basic syntax of an ALTER TABLE command to add a **New Column** in an existing table is as follows.

```
ALTER TABLE table_name ADD column_  
name datatype;
```

The basic syntax of an ALTER TABLE command to **DROP COLUMN** in an existing table is as follows.

```
ALTER TABLE table_name DROP COLUMN  
column_name;
```

The basic syntax of an ALTER TABLE command to change the **DATA TYPE** of a column in a table is as follows.

```
ALTER TABLE table_name MODIFY  
COLUMN column_name datatype;
```

The basic syntax of an ALTER TABLE command to add a **NOT NULL** constraint to a column in a table is as follows.

ALTER TABLE table\_name MODIFY column \_name datatype NOT NULL;

The basic syntax of ALTER TABLE to **ADD UNIQUE CONSTRAINT** to a table is as follows.

ALTER TABLE table\_name

ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);

The basic syntax of an ALTER TABLE command to **ADD CHECK CONSTRAINT** to a table is as follows.

ALTER TABLE table\_name

ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);

The basic syntax of an ALTER TABLE command to **ADD PRIMARY KEY** constraint to a table is as follows.

ALTER TABLE table\_name

ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

The basic syntax of an ALTER TABLE command to **DROP CONSTRAINT** from a table is as follows.

ALTER TABLE table\_name

DROP CONSTRAINT MyUniqueConstraint;

If you're using MySQL, the code is as follows "

ALTER TABLE table\_name

DROP INDEX MyUniqueConstraint;

The basic syntax of an ALTER TABLE command to **DROP PRIMARY KEY** constraint from a table is as follows.

ALTER TABLE table\_name

DROP CONSTRAINT MyPrimaryKey;

If you're using MySQL, the code is as follows "

ALTER TABLE table\_name

DROP PRIMARY KEY;

### Example

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	1000.00

Following is the example to ADD a **New Column** to an existing table "

```
ALTER TABLE CUSTOMERS ADD SEX char(1);
```

Now, the CUSTOMERS table is changed and following would be output from the SELECT statement.

ID	NAME	AGE	ADDRESS	SALARY	SEX
1	Ramesh	32	Ahmedabad	2000.00	NULL
2	Ramesh	25	Delhi	1500.00	NULL
3	Kaushik	23	Kota	2000.00	NULL
4	Kaushik	25	Mumbai	6500.00	NULL
5	Hardik	27	Bhopal	8500.00	NULL
6	Komal	22	MP	4500.00	NULL
7	Muffy	24	Indore	10000.00	NULL

Following is the example to DROP sex column from the existing table.

```
ALTER TABLE CUSTOMERS DROP SEX;
```

Now, the CUSTOMERS table is changed and following would be the output from the SELECT statement.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Ramesh	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Kaushik	25	Mumbai	6500.00
5.	Hardik	27	Bhopal	8500.00
6.	Komal	22	MP	4500.00
7.	Muffy	24	Indore	10000.00

#### 4. DROP

The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

##### NOTE

You should be very careful while using this command because once a table is deleted then all the information available in that table will also be lost forever.

##### Syntax

The basic syntax of this DROP TABLE statement is as follows:

```
DROP TABLE table_name;
```

##### Example

Let us first verify the CUSTOMERS table and then we will delete it from the database as shown below:

```
SQL> DESC CUSTOMERS;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI		
NAME	varchar(20)	NO			
AGE	int(11)	NO			
ADDRESS	char(25)	YES	NULL		
SALARY	decimal(18,2)	YES	NULL		

5 rows inset(0.00 sec)

This means that the CUSTOMERS table is available in the database, so let us now drop it as shown below.

SQL> DROP TABLE CUSTOMERS;

Query OK,0 rows affected (0.01 sec)

Now, if you would try the DESC command, then you will get the following error "

SQL> DESC CUSTOMERS;

## OUTPUT

ERROR 1146(42S02):Table'TEST.CUSTOMERS' doesn't exist

## 5. TRUNCATE

The SQL TRUNCATE TABLE command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure from the database and you would need to re-create this table once again if you wish you store some data.

## Syntax

The basic syntax of a TRUNCATE TABLE command is as follows.

TRUNCATE TABLE table\_name;

## Example

Consider a CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is the example of a Truncate command.

SQL > TRUNCATE TABLE CUSTOMERS;

Now, the CUSTOMERS table is truncated and the output from SELECT statement will be as shown in the code block below:

```
SQL> SELECT * FROM CUSTOMERS;
```

**OUTPUT:**

Empty set (0.00 sec)

**6. RENAME**

SQL RENAME Statement

With RENAME statement you can rename a table.

**Syntax for SQL RENAME is:**

```
RENAME TABLE {tbl_name} TO {new_tbl_name};
```

**Where {tbl\_name} table that exists in the current database, and {new\_tbl\_name} is new table name.**

```
ALTER TABLE {tbl_name} RENAME TO {new_tbl_name};
```

As Example

```
CREATE TABLE employees
```

```
( id NUMBER(6), name VARCHAR(20)
```

```
);
```

```
INSERT INTO employees( id, name ) values( 1, 'name 1');
```

```
INSERT INTO employees( id, name ) values( 2, 'name 2');
```

```
INSERT INTO employees( id, name ) values( 3, 'name 3');
```

```
SELECT * FROM employees;
```

**SELECT Output**

id	Name
1	name 1
2	name 2
3	name 3

```
RENAME TABLE employees TO employees_new;
```

```
SELECT * FROM employees_new;
```

**SELECT Output**

id	Name
1	name 1
2	name 2
3	name 3

**Q8. Explain in detail about DML commands ?**

*Ans :*

**DML(Data Manipulation Language)**

The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

**Examples of DML**

- **SELECT** is used to retrieve data from the a database.
- **INSERT** is used to insert data into a table.
- **UPDATE** is used to update existing data within a table.
- **DELETE** is used to delete records from a database table.

**1. INSERT**

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

**Syntax**

There are two basic syntaxes of the INSERT INTO statement which are shown below.

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
```

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The **SQL INSERT INTO** syntax will be as follows:

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

**Example**

The following statements would create six records in the CUSTOMERS table.

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1,'Ramesh',32,'Ahmedabad',2000.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2,'Khilan',25,'Delhi',1500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3,'kaushik',23,'Kota',2000.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4,'Chaitali',25,'Mumbai',6500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5,'Hardik',27,'Bhopal',8500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6,'Komal',22,'MP',4500.00);
```



You can create a record in the CUSTOMERS table by using the second syntax as shown below.

INSERT INTO CUSTOMERS

VALUES (7, 'Muffy', 24, 'Indore', 10000.00 );

All the above statements would produce the following records in the CUSTOMERS table as shown below.

ID	NAME	AGE	ADDRESS	SALARY
1.	Ramesh	32	Ahmedabad	2000.00
2.	Khilan	25	Delhi	1500.00
3.	Kaushik	23	Kota	2000.00
4.	Chaitali	25	Mumbai	6500.00
5.	Hardik	27	Bhopal	8500.00
6.	Komal	22	MP	4500.00
7.	Muffy	24	Indore	10000.00

## 2. SELECT

The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

### Syntax

The basic syntax of the SELECT statement is as follows:

SELECT column1, column2, columnN FROM table\_name;

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

SELECT \* FROM table\_name;

### Example

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code is an example, which would fetch the ID, Name and Salary fields of the customers available in CUSTOMERS table.

SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS;

This would produce the following result:

ID	NAME	SALARY
1	Ramesh	2000.00
2	Khilan	1500.00
3	Kaushik	2000.00
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	Muffy	10000.00

If you want to fetch all the fields of the CUSTOMERS table, then you should use the following query.

```
SQL> SELECT * FROM CUSTOMERS;
```

This would produce the result as shown below.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

### 3. UPDATE

The SQL **UPDATE** Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

#### Syntax

The basic syntax of the UPDATE query with a WHERE clause is as follows "

```
UPDATE table_name
```

```
SET column1 = value1, column2 = value2..., columnN = valueN
```

```
WHERE [condition];
```

You can combine N number of conditions using the AND or the OR operators.

**Example**

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following query will update the ADDRESS for a customer whose ID number is 6 in the table.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune'
WHERE ID =6;
```

Now, the CUSTOMERS table would have the following records :

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

If you want to modify all the ADDRESS and the SALARY column values in the CUSTOMERS table, you do not need to use the WHERE clause as the UPDATE query would be enough as shown in the following code block.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune', SALARY =1000.00;
```

Now, CUSTOMERS table would have the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Pune	1000.00
2	Khilan	25	Pune	1000.00
3	Kaushik	23	Pune	1000.00
4	Chaitali	25	Pune	1000.00
5	Hardik	27	Pune	1000.00
6	Komal	22	Pune	1000.00
7	Muffy	24	Pune	1000.00

## DELETE

The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

### Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows "

```
DELETE FROM table_name
```

```
WHERE [condition];
```

You can combine N number of conditions using AND or OR operators.

### Example

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code has a query, which will DELETE a customer, whose ID is 6.

```
SQL> DELETE FROM CUSTOMERS
```

```
WHERE ID =6;
```

Now, the CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

If you want to DELETE all the records from the CUSTOMERS table, you do not need to use the WHERE clause and the DELETE query would be as follows "

```
SQL> DELETE FROM CUSTOMERS;
```

Now, the CUSTOMERS table would not have any record.

**Q9. Explain in detail about DCL Commands***Ans. :***(Imp.)**

Data Control Language(DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges. Privileges are of two types,

- **System:** This includes permissions for creating session, table, etc and all types of other system privileges.
- **Object:** This includes permissions for any command or query to perform any operation on the database tables.

In DCL we have two commands,

1. **GRANT:** Used to provide any user access privileges or other privileges for the database.
2. **REVOKE:** Used to take back permissions from any user.

**1. GRANT COMMAND**

- **GRANT command** gives user's access privileges to the database.
- This command allows specified users to perform specific tasks.

**Syntax:**

GRANT <privilege list>

ON <relation name or view name>

TO <user/role list>;

**Example :** GRANT Command

GRANT ALL ON employee

TO ABC;

[WITH GRANT OPTION]

In the above example, user 'ABC' has been given permission to view and modify the records in the 'employee' table.

**2. REVOKE COMMAND**

- **REVOKE command** is used to cancel previously granted or denied permissions.
- This command withdraw access privileges given with the GRANT command.
- It takes back permissions from user.

**Syntax:**

REVOKE <privilege list>

ON <relation name or view name>

FROM <user name>;

**Example :** REVOKE Command

REVOKE UPDATE

ON employee

FROM ABC;

**Q10. What are the differences between grant and revoke***Ans :*

GRANT	REVOKE
<p><b>GRANT command</b> allows a user to perform certain activities on the database.</p> <p>It grants access privileges for database objects to other users.</p> <p><b>Example:</b></p> <pre>GRANT privilege_name ON object_name TO {     user_name   PUBLIC   role_name } [WITH GRANT OPTION];</pre>	<p><b>REVOKE command</b> disallows a user to perform certain activities.</p> <p>It revokes access privileges for database objects previously granted to other users.</p> <p><b>Example:</b></p> <pre>REVOKE privilege_name ON object_name FROM {     user_name   PUBLIC   role_name }</pre>

**3.1.3 Basic structure of SQL Queries****Q11. What is the basic structure of SQL commands.***Ans :*

A relational database is a collection of tables. Each table has its own unique name.

The basic structure of an SQL expression consists of three clauses:

- The **select** clause which corresponds to the projection operation. It is the list of attributes that will appear in the resulting table.
- The **from** clause which corresponds to the Cartesian-product operation. It is the list of tables that will be joined in the resulting table.
- The **where** clause which corresponds to the selection operation. It is the expression that controls the which rows appear in the resulting table.

A typical SQL query has the form of:

```
select    A1, A2, ..., An
from      r1, r2, ..., rn,
where    P
```

The query is the equivalent to the relational algebra expression

$$\rho_{A1, A2, \dots, An} (\sigma_P(r_1 \bowtie r_2 \bowtie \dots \bowtie r_n))$$
**The select Clause**

Formal query languages are based on the mathematical notion of a relation being a set. Duplicate tuples never appear in relations. In practice, duplicate elimination is relatively time consuming. SQL allows duplicates in relations as well as the results of SQL expressions.

In those cases where we want to force the elimination of duplicates, we insert the keyword **distinct** after **select**. The default is to retain duplicates. This can be explicitly required with the keyword **all**.

The asterisk symbol "\*" can be used in place of listing all the attributes.

The clause can also contain arithmetic expressions involving the operators +, -, \*, and /.

A dot notation is used when explicitly identifying the table that the attribute comes from: **borrower.loan-number**

The **from** Clause

The **from** clause defines a Cartesian product of the tables in the clause.

The **where** Clause

SQL uses **and**, **or** and **not** (not symbols) and the comparison operators <, <=, >, >=, =, and <>. Also available is **between**:

**where amount between 90000 and 100000**

Additional, **not between** can be used.

**The rename Operation**

**SQL uses the as clause:**

*old\_name as new\_name*

You can do pattern matching on strings, using **like** and special characters:

- % which matching any substring
- \_ which matches any single character
- **escape** which allows you override another character:
- **like** "ab\%cd%" **escape** "\" which matches any string that starts with "ab\cd"

### Ordering the Display of Tuples

SQL uses the **order by** clause to control the order of the display of rows, either ascending (**asc**) or descending (**desc**):

**order by amount desc**

Sorting a large number of tuples may be costly and its use should be limited.

### Q12. How to use expressions in the WHERE Clause?

*Ans :*

Expressions can also be used in the WHERE clause of the SELECT statement.

**For example:**

Lets consider the employee table. If you want to display employee name, current salary, and a 20% increase in the salary for only those products where the percentage increase in salary is greater than 30000, the SELECT statement can be written as shown below

```
SELECT name, salary, salary*1.2 AS new_salary FROM employee
WHERE salary*1.2 > 30000;
```

**Output**

name	salary	new_salary
Hrithik	35000	37000
Harsha	35000	37000
Priya	30000	360000

**Q13. What is the use of Group by clause in SQL ?***Ans :***SQL GROUP BY Clause**

The SQL GROUP BY Clause is used along with the group functions to retrieve data grouped according to one or more columns.

**For Example:** If you want to know the total amount of salary spent on each department, the query would be:

```
SELECT dept, SUM (salary)
```

```
FROM employee
```

```
GROUP BY dept;
```

The output would be like:

dept	salary
Electrical	25000
Electronics	55000
Aeronautics	35000
InfoTech	30000

**Note:**

The group by clause should contain all the columns in the select list except those used along with the group functions.

```
SELECT location, dept, SUM (salary)
```

```
FROM employee
```

```
GROUP BY location, dept;
```

The output would be like:

location	dept	salary
Bangalore	Electrical	25000
Bangalore	Electronics	55000
Mysore	Aeronautics	35000
Mangalore	InfoTech	30000

**Q14. What is the use of Having clause in SQL?***Ans :***SQL HAVING Clause**

Having clause is used to filter data based on the group functions. This is similar to WHERE condition but is used with group functions. Group functions cannot be used in WHERE Clause but can be used in HAVING clause.

**SQL HAVING Clause Example**

If you want to select the department that has total salary paid for its employees more than 25000, the sql query would be like;

```
SELECT dept, SUM (salary)
```

```
FROM employee
```

```
GROUP BY dept
```

```
HAVING SUM (salary) > 25000
```

The output would be like:

dept	salary
Electronics	55000
Aeronautics	35000
InfoTech	30000

When WHERE, GROUP BY and HAVING clauses are used together in a SELECT statement, the WHERE clause is processed first, then the rows that are returned after the WHERE clause is executed are grouped based on the GROUP BY clause.

Finally, any conditions on the group functions in the HAVING clause are applied to the grouped rows before the final output is displayed.

**3.1.4 Additional basic operations****Q15. Explain the use of as clause in SQL.***Ans :***(Imp.)**

- SQL '**AS**' is used to assign a new name temporarily to a table column or even a table.
- It makes an easy presentation of query results and allows the developer to label results more accurately without permanently renaming table columns or even the table itself.
- Let's see the syntax of select as:



- SELECT** Column\_Name1 **AS** New\_Column\_Name, Column\_Name2 **As** New\_Column\_Name **FROM** Table\_Name;

Here, the Column\_Name is the name of a column in the original table, and the New\_Column\_Name is the name assigned to a particular column only for that specific query. This means that New\_Column\_Name is a temporary name that will be assigned to a query.

#### Assigning a temporary name to the column of a table:

Let us take a table named orders, and it contains the following data:

Day_of_order	Customer	Product	Quantity
11-09-2001	Ajeet	Mobile	2
13-12-2001	Mayank	Laptop	20
26-12-2004	Balaswamy	Water cannon	35

#### Example:

Suppose you want to rename the 'day\_of\_order' column and the 'customer' column as 'Date' and 'Client', respectively.

#### Query:

- SELECT** day\_of\_order **AS** 'Date', Customer **As** 'Client', Product, Quantity **FROM** orders;

The result will be shown as this table:

Day_of_order	Customer	Product	Quantity
11-09-2001	Ajeet	Mobile	2
13-12-2001	Mayank	Laptop	20
26-12-2004	Balaswamy	Water cannon	35

From the above results, we can see that temporarily the 'Day\_of\_order' is renamed as 'date' and 'customer' is renamed as 'client'.

Let us take another example. Consider we have a students table with the following data.

Student_RollNo	Student_Name	Student_Gender	Student_Mobile Number	Student_Home Town	Student_Age	Student_Percentage
1	Rohit More	Male	9890786123	Lucknow	23	75
2	Kunal Shah	Male	7789056784	Chandigarh	20	92
3	Kartik Goenka	Male	9908743576	Ahemdabad	22	89
4	Anupama Shah	Female	8890907656	Chennai	24	92
5	Snehal Jain	Female	8657983476	Surat	21	94

#### Example 1

Write a query to get the student name and the average of the percentage of the student under the temporary column name 'Student' and 'Student\_Percentage', respectively.

#### Query:

- SELECT** Student\_Name **AS** Student, AVG (Student\_Percentage) **AS** Average\_Percentage **FROM** students;

Here, to calculate the average, we have used **AVG () function**. Further, the calculated average value of the percentage will be stored under the temporary name 'Average\_Percentage'.

The result will be shown as this table:

Student	Average_Percentage
Rohit More	88.4000

## Example 2

Write a query to get the student roll number and the student mobile number under the temporary column name 'Roll No' and 'Mobile Number', respectively.

### Query

- mysql> **SELECT** Student\_RollNo **AS** 'Roll No', Student\_PhoneNumber **AS** 'Mobile Number'  
**FROM** students;

The result will be shown as this table:

Roll No	Mobile Number
1	9890786123
2	7789056784
3	9908743576
4	8890907656
5	8657983476

## Q16. What is the use of Rename command in SQL ?

*Ans :*

In some situations, database administrators and users want to change the name of the table in the SQL database because they want to give a more relevant name to the table.

Any database user can easily change the name by using the RENAME TABLE and ALTER TABLE statement in Structured Query Language.

The RENAME TABLE and ALTER TABLE syntax help in changing the name of the table.

### Syntax of RENAME statement in SQL

- RENAME old\_table \_name To new\_table\_name;

### Examples of RENAME statement in SQL

Here, we have taken the following two different SQL examples, which will help you how to change the name of the SQL table in the database using RENAME statement:

**Example 1:** Let's take an example of a table named **Cars**:

Car Name	Car Color	Car Cost
Hyundai Creta	White	10,85,000
Hyundai Venue	White	9,50,000
Hyundai i20	Red	9,00,000
Kia Sonet	White	10,00,000
Kia Seltos	Black	8,00,000
Swift Dezire	Red	7,95,000

**Table: Cars**

- Suppose, you want to change the above table name into "Car\_2021\_Details". For this, you have to type the following RENAME statement in SQL:
  1. RENAME Cars To Car\_2021\_Details ;
- After this statement, the table "Cars" will be changed into table name "Car\_2021\_Details".

**Q17. What are the basic string operations of SQL ?***Ans :*

The string operators in SQL are used to perform important operations such as pattern matching, concatenation, etc. where pattern matching is performed by using the wildcard characters such as '%' and '\_' in concurrence with the Like operator to search for the specific patterns in strings and by the usage of concatenation operation one or more strings or columns of the tables can be combined together.

The concatenation of strings, as well as pattern matching, can be performed by using the below operators in SQL. Let us look at a few examples.

**1. Concatenation Operator**

The concatenation operation is used to combine character strings, columns of a table or it can also be used for the combination of column and strings.

In the below example, we can see that the two strings 'Hello' and 'World!' are combined by using the '+' in between the string values.

```
SELECT 'Hello' + 'World!' AS StringConcatenated;
```

The above statement gives the below result where both the strings are combined and the same is displayed in the output.

	StringConcatenated
1	HelloWorld!

In the below statement, it can be seen that the two strings are concatenated along with space in between.

```
SELECT 'Hello' + ' ' + 'World!' AS StringConcatenated;
```

In the result below, the concatenation is performed where both the strings 'Hello' and 'World!' are combined along with a space in between them.

	StringConcatenated
1	Hello World!

In the above examples, we could see that the concatenation operation is performed and '+' is used along with the strings to combine strings as well as space between the strings values. The concatenation can be performed on the columns of the table also.

Let us take the example of the table "STUDENTS" as shown below.

select \* from STUDENTS;

	ROLL	FIRSTNAME	LASTNAME	AGE
1	18	Preety	Jha	21
2	19	Raj	Kumar	22
3	20	Harry	Smith	21

In the below example, the columns FIRSTNAME and LASTNAME of the table "STUDENTS" are combined with space in between the columns.

SELECT FIRSTNAME + ' ' + LASTNAME AS ConcatenatedName FROM STUDENTS;

In the below result, we can see that the FIRSTNAME and LASTNAME and the space in between them are concatenated.

	ConcatenatedName
1	Preety Jha
2	Raj Kumar
3	Harry Smith

## 2. Like Operator

This operator is used to decide if the specific character string matches the specific pattern where the pattern can be a regular or wildcard character. While pattern matching, the regular characters should match exactly with the specific characters of the string but when we want to match the arbitrary fragments of the string, wildcard characters can be used.

Let us take the example of the below query.

SELECT \* FROM STUDENTS WHERE FIRSTNAME='Preety';

The result of the above query is as shown below.

	ROLL	FIRSTNAME	LASTNAME	AGE
1	18	Preety	Jha	21

Here if we need to retrieve the details of the STUDENT Preeti, we need to remember the complete FIRSTNAME. In some other cases it might occur that it is not possible to remember the FIRSTNAME easily and in such cases the usage of pattern matching is helpful so that the retrieval of the data is possible even with a partial match of the student's first name. The usage of Like in the below query will check if the data value in the column matches with the specific pattern. Here the pattern may also include the wildcard characters. The wildcard character used in the below query is % and any sequence of zero or more characters are matched by %.

So the previous query is now altered as below with the usage of Like along with the wildcard character.

```
SELECT * FROM STUDENTS WHERE FIRSTNAME LIKE 'p%';
```

In the above query, the FIRSTNAME column is compared with the pattern 'p%' and then it finds the Student name which starts with 'p' as shown below.

	ROLL	FIRSTNAME	LASTNAME	AGE
1	18	Preety	Jha	21

In the below query, it can be seen that the wildcard character % is used before 'j' and this will find the values which end with 'j'.

```
SELECT * FROM STUDENTS WHERE FIRSTNAME LIKE '%j';
```

The result of the above statement below shows the output of the Student name which ends with 'j'.

	ROLL	FIRSTNAME	LASTNAME	AGE
1	19	Raj	Kumar	22

### 3.1.5 Set Operations

**Q18. List and explain various set operations of SQL.**

*Ans .:*

**(Imp.)**

SQL supports few Set operations which can be performed on the table data. These are used to get meaningful results from data stored in the table, under different special conditions.

In this tutorial, we will cover 4 different types of SET operations, along with example:

1. UNION
2. UNION ALL
3. INTERSECT
4. MINUS

#### 1. Union

- The SQL Union operation is used to combine the result of two or more SQL SELECT queries.
- In the union operation, all the number of datatype and columns must be same in both the tables on which UNION operation is being applied.
- The union operation eliminates the duplicate rows from its resultset.

#### Syntax

1. SELECT column\_name FROM table1
2. UNION
3. SELECT column\_name FROM table2;

**Example:****The First table**

ID	NAME
1	Jack
2	Harry
3	Jackson

**The Second table**

ID	NAME
3	Jackson
4	Stephan
5	David

**Union SQL query will be:**

1. SELECT \* FROM First
2. UNION
3. SELECT \* FROM Second;

The resultset table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
4	Stephan
5	David

**2. Union All**

Union All operation is equal to the Union operation. It returns the set without removing duplication and sorting the data.

**Syntax:**

1. SELECT column\_name FROM table1
2. UNION ALL
3. SELECT column\_name FROM table2;

**Example:** Using the above First and Second table.

Union All query will be like:

1. SELECT \* FROM First
2. UNION ALL
3. SELECT \* FROM Second;

The resultset table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
3	Jackson
4	Stephan
5	David

**3. Intersect**

- It is used to combine two SELECT statements. The Intersect operation returns the common rows from both the SELECT statements.
- In the Intersect operation, the number of datatype and columns must be the same.
- It has no duplicates and it arranges the data in ascending order by default.

**Syntax**

1. SELECT column\_name FROM table1
2. INTERSECT
3. SELECT column\_name FROM table2;

**Example:**

Using the above First and Second table.

Intersect query will be:

1. SELECT \* FROM First
2. INTERSECT
3. SELECT \* FROM Second;

The resultset table will look like:

ID	NAME
3	Jackson

**4. Minus**

- It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- It has no duplicates and data arranged in ascending order by default.

**Syntax:**

1. SELECT column\_name FROM table1
2. MINUS
3. SELECT column\_name FROM table2;

**Example**

Using the above First and Second table.

Minus query will be:

1. SELECT \* FROM First
2. MINUS
3. SELECT \* FROM Second;

The resultset table will look like:

ID	NAME
1	Jack
2	Harry

**3.1.6 Null values****Q19. What is the use of Null values?**

*Ans :*

The SQL **NULL** is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank.

A field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces.

**Syntax**

The basic syntax of **NULL** while creating a table.

```
SQL> CREATE TABLE CUSTOMERS(
    ID                INT                NOT NULL,
    NAME              VARCHAR (20)      NOT NULL,
    AGE               INT                NOT NULL,
    ADDRESS            CHAR (25),
    SALARY             DECIMAL (18, 2),
    PRIMARY KEY (ID)
);
```

Here, **NOT NULL** signifies that column should always accept an explicit value of the given data type. There are two columns where we did not use NOT NULL, which means these columns could be NULL.

A field with a NULL value is the one that has been left blank during the record creation.

**Example**

The NULL value can cause problems when selecting data. However, because when comparing an unknown value to any other value, the result is always unknown and not included in the results. You must use the **IS NULL** or **IS NOT NULL** operators to check for a NULL value.

Consider the following CUSTOMERS table having the records as shown below.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5.	Hardik	27	Bhopal	8500.00
6.	Komal	22	MP	4500.00
7.	Muffy	24	Indore	10000.00

Now, following is the usage of the **IS NOT NULL** operator.

```
SQL> SELECT ID, NAME, AGE, ADDRESS, SALARY
FROM CUSTOMERS
WHERE SALARY IS NOT NULL;
```

This would produce the following result:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5.	Hardik	27	Bhopal	8500.00

Now, following is the usage of the **IS NULL** operator.

```
SQL> SELECT ID, NAME, AGE, ADDRESS, SALARY
FROM CUSTOMERS
WHERE SALARY IS NULL;
```

This would produce the following result:

ID	NAME	AGE	ADDRESS	SALARY
6.	Komal	22	MP	4500.00
7.	Muffy	24	Indore	10000.00



### 3.1.7 Aggregate functions

**Q20. Explain the concept of various aggregate functions.**

*Ans :*

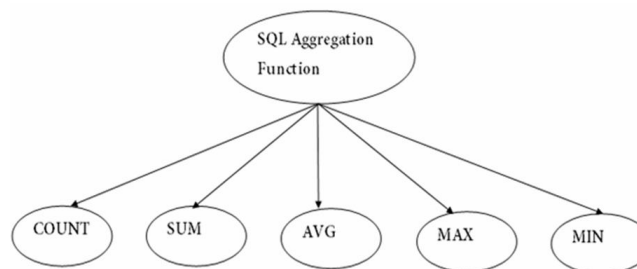
(Imp.)

SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.

Aggregate operators (functions) improves the data retrieval property of SQL statement. These operators enables the users to summarize the data retrieved from the relations. Basically, the aggregate operators takes the entire column of data as its argument and generates a result set that summarizes the column. The following are the different aggregate operators supported by SQL.

It is also used to summarize the data.

#### Types of SQL Aggregation Function



#### 1. Count Function

- COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
- COUNT function uses the COUNT(\*) that returns the count of all the rows in a specified table. COUNT(\*) considers duplicate and Null.

#### Syntax

COUNT(\*)

or

COUNT( [ALL|DISTINCT] expression)

#### Sample table

#### PRODUCT\_MAST

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Cpm1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50
Item10	Com3	4	30	120

**Example: COUNT()**

```
SELECT COUNT(*)
FROM PRODUCT_MAST;
```

**Output:**

10

**Example: COUNT with WHERE**

```
SELECT COUNT(*)
FROM PRODUCT_MAST;
WHERE RATE >= 20;
```

**Output:**

7

**Example: COUNT() with DISTINCT**

```
SELECT COUNT(DISTINCT COMPANY)
FROM PRODUCT_MAST;
```

**Output:**

3

**Example: COUNT() with GROUP BY**

```
SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY;
```

**Output:**

```
Com1  5
Com2  3
Com3  2
```

**Example: COUNT() with HAVING**

```
SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING COUNT(*) > 2;
```

**Output:**

```
Com1  5
Com2  3
```

**2. SUM Function**

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

**Syntax**

```
SUM()
or
SUM( [ALL|DISTINCT] expression )
```

**Example: SUM()**

```
SELECT SUM(COST)
FROM PRODUCT_MAST;
```

**Output**

670

**Example: SUM() with WHERE**

```
SELECT SUM(COST)
FROM PRODUCT_MAST
WHERE QTY > 3;
```

**Output:**

320

**Example: SUM() with GROUP BY**

```
SELECT SUM(COST)
FROM PRODUCT_MAST
WHERE QTY > 3
GROUP BY COMPANY;
```

**Output:**

```
Com1  150
Com2  170
```

**Example: SUM() with HAVING**

```
SELECT COMPANY, SUM(COST)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING SUM(COST) >= 170;
```

**Output:**

```
Com1  335
Com3  170
```

**3. AVG function**

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

**Syntax**

AVG()

or

AVG( [ALL|DISTINCT] expression )

**Example:**

```
SELECT AVG(COST)
FROM PRODUCT_MAST;
```

**Output:**

67.00

**4. MAX Function**

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

**Syntax**

MAX()

or

MAX( [ALL|DISTINCT] expression )

**Example:**

```
SELECT MAX(RATE)
FROM PRODUCT_MAST;
30
```

**5. MIN Function**

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

**Syntax**

MIN()

or

MIN( [ALL|DISTINCT] expression )

**Example:**

```
SELECT MIN(RATE)
FROM PRODUCT_MAST;
```

**Output:**

10

**3.1.8 Nested subqueries****Q21. What is the nested sub query in SQL ?**

(OR)

**How many Subqueries can be nested in SQL?**

*Ans :*

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow:

- A subquery can be placed in a number of SQL clauses like WHERE clause, FROM clause, HAVING clause.
- You can use Subquery with SELECT, UPDATE, INSERT, DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
- A subquery is a query within another query. The outer query is known as the main query, and the inner query is known as a subquery.
- Subqueries are on the right side of the comparison operator.
- A subquery is enclosed in parentheses.
- In the Subquery, ORDER BY command cannot be used. But GROUP BY command can be used to perform the same function as ORDER BY command.

**1. Subqueries with the Select Statement**

SQL subqueries are most frequently used with the Select statement.

**Syntax**

Exception Handling in Java - Javatpoint

1. SELECT column\_name

2. FROM table\_name
3. WHERE column\_name expression operator
4. (SELECT column\_name from table\_name WHERE ... );

### Example

Consider the EMPLOYEE table have the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	John	20	US	2000.00
2	Stephan	26	Dubai	1500.00
3	David	27	Bangkok	2000.00
4	Alina	29	UK	6500.00
5	Kathrin	34	Bangalore	8500.00
6	Harry	42	China	4500.00
7	Jackson	25	Mizoram	10000.00

The subquery with a SELECT statement will be:

1. SELECT \*
2. FROM EMPLOYEE
3. WHERE ID IN (SELECT ID
4. FROM EMPLOYEE
5. WHERE SALARY>4500);

This would produce the following result:

ID	NAME	AGE	ADDRESS	SALARY
4	Alina	29	UK	6500.00
5	Kathrin	34	Bangalore	8500.00
7	Jackson	25	Mizoram	10000.00

## 2. Subqueries with the INSERT Statement

- SQL subquery can also be used with the Insert statement. In the insert statement, data returned from the subquery is used to insert into another table.
- In the subquery, the selected data can be modified with any of the character, date functions.

### Syntax

1. INSERT INTO table\_name (column1, column2, column3....)
2. SELECT
3. FROM table\_name
4. WHERE VALUE OPERATOR

**Example**

Consider a table EMPLOYEE\_BKP with similar as EMPLOYEE.

Now use the following syntax to copy the complete EMPLOYEE table into the EMPLOYEE\_BKP table.

1. INSERT INTO EMPLOYEE\_BKP
2. SELECT \* FROM EMPLOYEE
3. WHERE ID IN (SELECT ID
4. FROM EMPLOYEE);

**3. Subqueries with the UPDATE Statement**

The subquery of SQL can be used in conjunction with the Update statement. When a subquery is used with the Update statement, then either single or multiple columns in a table can be updated.

**Syntax**

1. UPDATE table
2. SET column\_name = new\_value
3. WHERE VALUE OPERATOR
4. (SELECT COLUMN\_NAME
5. FROM TABLE\_NAME
6. WHERE condition);

**Example**

Let's assume we have an EMPLOYEE\_BKP table available which is backup of EMPLOYEE table. The given example updates the SALARY by .25 times in the EMPLOYEE table for all employee whose AGE is greater than or equal to 29.

1. UPDATE EMPLOYEE
2. SET SALARY = SALARY \* 0.25
3. WHERE AGE IN (SELECT AGE FROM CUSTOMERS\_BKP
4. WHERE AGE >= 29);

This would impact three rows, and finally, the EMPLOYEE table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	John	20	US	2000.00
2	Stephan	26	Dubai	1500.00
3	David	27	Bangkok	2000.00
4	Alina	29	UK	1625.00
5	Kathrin	34	Bangalore	2125.00
6	Harry	42	China	1125.00
7	Jackson	25	Mizoram	10000.00

#### 4. Subqueries with the DELETE Statement

The subquery of SQL can be used in conjunction with the Delete statement just like any other statements mentioned above.

##### Syntax

1. DELETE FROM TABLE\_NAME
2. WHERE VALUE OPERATOR
3. (SELECT COLUMN\_NAME
4. FROM TABLE\_NAME
5. WHERE condition);

##### Example

Let's assume we have an EMPLOYEE\_BKP table available which is backup of EMPLOYEE table. The given example deletes the records from the EMPLOYEE table for all EMPLOYEE whose AGE is greater than or equal to 29.

1. DELETE FROM EMPLOYEE
2. WHERE AGE IN (SELECT AGE FROM EMPLOYEE\_BKP
3. WHERE AGE >= 29);

This would impact three rows, and finally, the EMPLOYEE table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	John	20	US	2000.00
2	Stephan	26	Dubai	1500.00
3	David	27	Bangkok	2000.00
7	Jackson	25	Mizoram	10000.00

#### 3.1.9 Modification of the database

##### Q22. How can we modify the database using SQL ?

*Ans :*

The SQL commands (UPDATE and DELETE) are used to modify the data that is already in the database. The SQL DELETE command uses a WHERE clause.

**SQL UPDATE** statement is used to change the data of the records held by tables. Which rows is to be update, it is decided by a condition. To specify condition, we use WHERE clause.

The UPDATE statement can be written in following form:

**UPDATE** table\_name **SET** [column\_name1 = value1,... column\_nameN = valueN] [**WHERE** condition]

##### Syntax

**UPDATE** table\_name  
**SET** column\_name = expression  
**WHERE** conditions

**Example:**

Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

**UPDATE Table**

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.

**Example**

```
UPDATE Customers
```

```
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
```

```
WHERE CustomerID = 1;
```

**Try it Yourself**

The selection from the "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

**Delete statement**

The DELETE statement is used to delete existing records in a table.

Syntax

```
DELETE FROM table_name WHERE condition;
```

The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:

**Example**

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

The "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

### 3.2 INTERMEDIATE SQL

#### 3.2.1 Join Expressions

**Q23. What is Join? List out various types of joins supported by SQL?**

*Ans :*

(Imp.)

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

a Consider the following two tables:

**Table 1: CUSTOMERS Table**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

**Table 2: ORDERS Table**

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-0800:00:00	3	3000
100	2009-10-0800:00:00	3	1500
101	2009-11-2000:00:00	2	1560
103	2008-05-2000:00:00	4	2060

Now, let us join these two tables in our SELECT statement as shown below.

```
SQL> SELECT ID, NAME, AGE, AMOUNT
FROM CUSTOMERS, ORDERS
WHERE CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```



This would produce the following result.

ID	NAME	AGE	AMOUNT
3	Kaushik	23	3000
3	Kaushik	23	1500
2	Khilan	25	1560
4	Chaitali	25	2060

Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal to symbol.

There are different types of joins available in SQL "

- **INNER JOIN:** returns rows when there is a match in both tables.
- **LEFT JOIN:** returns all rows from the left table, even if there are no matches in the right table.
- **RIGHT JOIN:** returns all rows from the right table, even if there are no matches in the left table.
- **FULL JOIN:** returns rows when there is a match in one of the tables.
- **SELF JOIN:** is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- **CARTESIAN JOIN:** returns the Cartesian product of the sets of records from the two or more joined tables.

**Q24. Explain inner join concept with an example.**

*Ans :*

**(Imp.)**

The most important and frequently used of the joins is the **INNER JOIN**. They are also referred to as an **EQUIJOIN**.

The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

### Syntax

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

SELECT table1.column1, table2.column2...

FROM table1

INNER JOIN table2

ON table1.common\_field = table2.common\_field;

### Example

Consider the following two tables.

Table 1: CUSTOMERS Table is as follows.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Table 2: ORDERS Table is as follows.

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-0800:00:00	3	3000
100	2009-10-0800:00:00	3	1500
101	2009-11-2000:00:00	2	1560
103	2008-05-2000:00:00	4	2060

Now, let us join these two tables using the INNER JOIN as follows

```
SQL> SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
INNER JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

This would produce the following result.

ID	NAME	AMOUNT	DATE
3	Kaushik	3000	2009-10-0800:00:00
3	Kaushik	1500	2009-10-0800:00:00
2	Khilan	1560	2009-11-2000:00:00
4	Chaitali	2060	2008-05-2000:00:00

**Q25. What is Left Join explain with an example.**

*Ans :*

**(Imp.)**

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

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

#### Syntax

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

SELECT table1.column1, table2.column2...

FROM table1

LEFT JOIN table2

ON table1.common\_field = table2.common\_field;

Here, the given condition could be any given expression based on your requirement.

### Example

Consider the following two tables,

**Table 1: CUSTOMERS Table is as follows.**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

**Table 2: Orders Table is as follows.**

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-0800:00:00	3	3000
100	2009-10-0800:00:00	3	1500
101	2009-11-2000:00:00	2	1560
103	2008-05-2000:00:00	4	2060

Now, let us join these two tables using the LEFT JOIN as follows.

```
SQL> SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
LEFT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

This would produce the following result:

ID	NAME	AMOUNT	DATE
3	Kaushik	3000	2009-10-0800:00:00
3	Kaushik	1500	2009-10-0800:00:00
2	Khilan	1560	2009-11-2000:00:00
4	Chaitali	2060	2008-05-2000:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL

**Q26. What is Right Join? Explain with an example.***Ans :*

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

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

**Syntax**

The basic syntax of a **RIGHT JOIN** is as follow.

```
SELECT table1.column1, table2.column2...
```

```
FROM table1
```

```
RIGHT JOIN table2
```

```
ON table1.common_field = table2.common_field;
```

**Example**

Consider the following two tables,

**Table 1: CUSTOMERS Table is as follows**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

**Table 2: ORDERS Table is as follows.**

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-0800:00:00	3	3000
100	2009-10-0800:00:00	3	1500
101	2009-11-2000:00:00	2	1560
103	2008-05-2000:00:00	4	2060

Now, let us join these two tables using the RIGHT JOIN as follows.

```
SQL> SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
RIGHT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

This would produce the following result:

ID	NAME	AMOUNT	DATE
3	Kaushik	3000	2009-10-0800:00:00
3	Kaushik	1500	2009-10-0800:00:00
2	Khilan	1560	2009-11-2000:00:00
4	Chaitali	2060	2008-05-2000:00:00

**Q27. What is Full join ? Explain with an example.**

*Ans :*

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

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

#### Syntax

The basic syntax of a **FULL JOIN** is as follows “

SELECT table1.column1, table2.column2...

FROM table1

FULL JOIN table2

ON table1.common\_field = table2.common\_field;

Here, the given condition could be any given expression based on your requirement.

#### Example

Consider the following two tables.

**Table 1: CUSTOMERS Table is as follows**

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

**Table 2: ORDERS Table is as follows**

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-0800:00:00	3	3000
100	2009-10-0800:00:00	3	1500
101	2009-11-2000:00:00	2	1560
103	2008-05-2000:00:00	4	2060

Now, let us join these two tables using FULL JOIN as follows.

```
SQL> SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
FULL JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

This would produce the following result:

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-200 0:00:00
3	Kaushik	3000	2009-10-080 0:00:00
3	Kaushik	1500	2009-10-080 0:00:00
4	Chaitali	2060	2008-05-200 0:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL
3	Kaushik	3000	2009-10-080 0:00:00
3	Kaushik	1500	2009-10-080 0:00:00
2	Khilan	1560	2009-11-200 0:00:00
4	Chaitali	2060	2008-05-200 0:00:00

If your Database does not support FULL JOIN (MySQL does not support FULL JOIN), then you can use **UNION ALL** clause to combine these two JOINS as shown below.

```
SQL> SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
LEFT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
UNION ALL
SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
RIGHT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
```

---

**Q28. What is self join ? Explain with an example?**

*Ans :*

The SQL **SELF JOIN** is used to join a table to itself as if the table were two tables; temporarily renaming at least one table in the SQL statement.

**Syntax**

The basic syntax of SELF JOIN is as follows:

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_
      field;
```

Here, the WHERE clause could be any given expression based on your requirement.

**Example**

Consider the following table.

**CUSTOMERS Table** is as follows.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, let us join this table using SELF JOIN as follows:

```
SQL> SELECT a.ID, b.NAME, a.SALARY
FROM CUSTOMERS a, CUSTOMERS b
WHERE a.SALARY < b.SALARY;
```

This would produce the following result:

ID	NAME	SALARY
2	Ramesh	1500.00
2	Kaushik	1500.00
1	Chaitali	2000.00
2	Chaitali	1500.00
3	Chaitali	2000.00
6	Chaitali	4500.00
1	Hardik	2000.00
2	Hardik	1500.00
3	Hardik	2000.00
4	Hardik	6500.00

6	Hardik	4500.00
1	Komal	2000.00
2	Komal	1500.00
3	Komal	2000.00
1	Muffy	2000.00
2	Muffy	1500.00
3	Muffy	2000.00
4	Muffy	6500.00
5	Muffy	8500.00
6	Muffy	4500.00

**3.2.2 Views**

**Q29. What is mean by view in SQL? Explain the process of creating, updating and dropping a view ?**

*Ans :* (Imp.)

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

**Creating Views**

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic **CREATE VIEW** syntax is as follows:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

You can include multiple tables in your SELECT statement in a similar way as you use them in a normal SQL SELECT query.

### Example

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example to create a view from the CUSTOMERS table. This view would be used to have customer name and age from the CUSTOMERS table.

```
SQL > CREATE VIEW CUSTOMERS_VIEW
AS
SELECT name, age
FROM CUSTOMERS;
```

Now, you can query CUSTOMERS\_VIEW in a similar way as you query an actual table. Following is an example for the same.

```
SQL > SELECT * FROM CUSTOMERS_
VIEW;
```

This would produce the following result.

name	age
Ramesh	32
Khilan	25
Kaushik	23
Chaitali	25
Hardik	27
Komal	22
Muffy	24

### The WITH CHECK OPTION

The with check option is a create view statement option. The purpose of the with check option is to ensure that all Update and Inserts satisfy the condition(s) in the view definition.

If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.

The following code block has an example of creating same view CUSTOMERS\_VIEW with the WITH CHECK OPTION.

```
CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM CUSTOMERS
WHERE age IS NOT NULL
WITH CHECK OPTION;
```

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.

### Updating a View

A view can be updated under certain conditions which are given below:

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.



- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if a view satisfies all the above-mentioned rules then you can update that view. The following code block has an example to update the age of Ramesh.

```
SQL > UPDATE CUSTOMERS_VIEW
      SET AGE = 35
      WHERE name = 'Ramesh';
```

This would ultimately update the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

#### Inserting Rows into a View

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

Here, we cannot insert rows in the CUSTOMERS\_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in a similar way as you insert them in a table.

#### Deleting Rows into a View

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE = 22.

```
SQL > DELETE FROM CUSTOMERS_VIEW
      WHERE age = 22;
```

This would ultimately delete a row from the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

#### Dropping Views

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below:

```
DROP VIEW view_name;
```

Following is an example to drop the CUSTOMERS\_VIEW from the CUSTOMERS table.

```
DROP VIEW CUSTOMERS_VIEW;
```

#### 3.2.3 Transactions

**Q30. What is transaction? Explain the properties of transaction ?**

*Ans :*

**(Imp.)**

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

A transaction is the propagation of one or more changes to the database. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing a transaction on that table. It is important to control these transactions to ensure the data integrity and to handle database errors.

Practically, you will club many SQL queries into a group and you will execute all of them together as a part of a transaction.

### Properties of Transactions

Transactions have the following four standard properties, usually referred to by the acronym **ACID**.

- **Atomicity:** Ensures that all operations within the work unit are completed successfully. Otherwise, the transaction is aborted at the point of failure and all the previous operations are rolled back to their former state.
- **Consistency:** Ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation:** Enables transactions to operate independently of and transparent to each other.
- **Durability:** Ensures that the result or effect of a committed transaction persists in case of a system failure.

### Q31. List out the commands used to control transactions?

*Ans :*

The following commands are used to control transactions.

- **COMMIT:** to save the changes.
- **ROLLBACK:** to roll back the changes.
- **SAVEPOINT:** creates points within the groups of transactions in which to ROLLBACK.
- **SET TRANSACTION:** Places a name on a transaction.

### Transactional Control Commands

Transactional control commands are only used with the **DML Commands** such as - INSERT, UPDATE and DELETE only. They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

### The COMMIT Command

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.

The syntax for the COMMIT command is as follows.

**COMMIT;**

### Example

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example which would delete those records from the table which have age = 25 and then COMMIT the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
```

```
WHERE AGE =25;
```

```
SQL> COMMIT;
```

Thus, two rows from the table would be deleted and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
3	Kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

**The ROLLBACK Command**

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

The syntax for a ROLLBACK command is as follows:

ROLLBACK;

**Example**

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is an example, which would delete those records from the table which have the age = 25 and then ROLLBACK the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
      WHERE AGE = 25;
```

```
SQL> ROLLBACK;
```

Thus, the delete operation would not impact the table and the SELECT statement would produce the following result.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

**The SAVEPOINT Command**

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 syntax for a SAVEPOINT command is as shown below.

SAVEPOINT SAVEPOINT\_NAME;

This command serves only in the creation of a SAVEPOINT among all the transactional statements. The ROLLBACK command is used to undo a group of transactions.

The syntax for rolling back to a SAVEPOINT is as shown below.

ROLLBACK TO SAVEPOINT\_NAME;

Following is an example where you plan to delete the three different records from the CUSTOMERS table. You want to create a SAVEPOINT before each delete, so that you can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state.

**Example**

Consider the CUSTOMERS table having the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code block contains the series of operations.

```
SQL> SAVEPOINT SP1;
```

Savepoint created.

```
SQL> DELETE FROM CUSTOMERS
      WHERE ID=1;
```

1 row deleted.

```
SQL> SAVEPOINT SP2;
```

Savepoint created.

```
SQL> DELETE FROM CUSTOMERS
WHERE ID=2;
```

1 row deleted.

```
SQL> SAVEPOINT SP3;
```

Savepoint created.

```
SQL> DELETE FROM CUSTOMERS
WHERE ID=3;
```

1 row deleted.

Now that the three deletions have taken place, let us assume that you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP2. Because SP2 was created after the first deletion, the last two deletions are undone:

```
SQL> ROLLBACK TO SP2;
```

Rollback complete.

Notice that only the first deletion took place since you rolled back to SP2.

```
SQL> SELECT * FROM CUSTOMERS;
```

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

6 rows selected

### The RELEASE SAVEPOINT Command

The RELEASE SAVEPOINT command is used to remove a SAVEPOINT that you have created.

The syntax for a RELEASE SAVEPOINT command is as follows.

```
RELEASE SAVEPOINT SAVEPOINT_NAME;
```

Once a SAVEPOINT has been released, you can no longer use the ROLLBACK command to undo transactions performed since the last SAVEPOINT.

### The SET TRANSACTION Command

The SET TRANSACTION command can be used to initiate a database transaction. This command is used to specify characteristics for the transaction that follows. For example, you can specify a transaction to be read only or read write.

The syntax for a SET TRANSACTION command is as follows.

```
SET TRANSACTION [ READ WRITE | READ
ONLY ];
```

### 3.2.4 Integrity constraints

**Q32. What is mean by constraints? List out most commonly used constraints in SQL?**

*Ans :* (Imp.)

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

Following are some of the most commonly used constraints available in SQL:

- **NOT NULL Constraint:** Ensures that a column cannot have NULL value.
- **DEFAULT Constraint:** Provides a default value for a column when none is specified.
- **UNIQUE Constraint:** Ensures that all values in a column are different.
- **PRIMARY Key:** Uniquely identifies each row/record in a database table.
- **FOREIGN Key:** Uniquely identifies a row/record in any of the given database table.
- **CHECK Constraint:** The CHECK constraint ensures that all the values in a column satisfies certain conditions.
- **INDEX:** Used to create and retrieve data from the database very quickly.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use the ALTER TABLE statement to create constraints even after the table is created.

**Q33. Discuss in detail about SQL integrity constraints**

*Ans :* (Imp.)

Integrity constraints are used to ensure accuracy and consistency of the data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity.

There are many types of integrity constraints that play a role in Referential Integrity (RI). These constraints include Primary Key, Foreign Key, Unique Constraints and other constraints

Integrity Constraints are used to apply business rules for the database tables.

The constraints available in SQL are

1. Primary key
2. Foreign Key
3. Not Null
4. Unique
5. Check.

**Constraints can be defined in two ways**

1. The constraints can be specified immediately after the column definition. This is called column-level definition.
2. The constraints can be specified after all the columns are defined. This is called table-level definition.

**1. Primary Key**

A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.

A table can have only one primary key, which may consist of single or multiple fields. When multiple fields are used as a primary key, they are called a composite key.

If a table has a primary key defined on any field(s), then you cannot have two records having the same value of that field(s).

**Note:**

You would use these concepts while creating database tables.

**Create Primary Key**

Here is the syntax to define the ID attribute as a primary key in a CUSTOMERS table.

```
CREATE TABLE CUSTOMERS(
    ID INT NOT NULL,
    NAME VARCHAR (20) NOT NULL,
    AGE INT NOT NULL,
    ADDRESS CHAR (25) ,
    SALARY DECIMAL (18, 2),
    PRIMARY KEY (ID)
);
```

**2. Foreign key**

A foreign key is a key used to link two tables together. This is sometimes also called as a referencing key.

A Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table.

The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

If a table has a primary key defined on any field(s), then you cannot have two records having the same value of that field(s).

**Example**

Consider the structure of the following two tables.

**CUSTOMERS table**

```
CREATE TABLE CUSTOMERS(
    ID INT NOT NULL,
    NAME VARCHAR (20) NOT NULL,
    AGE INT NOT NULL,
    ADDRESS CHAR (25),
    SALARY DECIMAL (18,2),
    PRIMARY KEY (ID)
);
```

**ORDERS table**

```
CREATE TABLE ORDERS (
    ID      INT      NOT NULL,
    DATE    DATETIME,
    CUSTOMER_ID INT references
                CUSTOMERS(ID),
    AMOUNT  double,
    PRIMARY KEY (ID)
);
```

If the ORDERS table has already been created and the foreign key has not yet been set, the use the syntax for specifying a foreign key by altering a table.

```
ALTER TABLE ORDERS
    ADD FOREIGN KEY (Customer_ID)
    REFERENCES CUSTOMERS (ID);
```

**DROP a FOREIGN KEY Constraint**

To drop a FOREIGN KEY constraint, use the following SQL syntax.

```
ALTER TABLE ORDERS
    DROP FOREIGN KEY;
```

**3. Not Null constraint**

By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such a constraint on this column specifying that NULL is now not allowed for that column.

A NULL is not the same as no data, rather, it represents unknown data.

**Example**

For example, the following SQL query creates a new table called CUSTOMERS and adds five columns, three of which, are ID NAME and AGE, In this we specify not to accept NULLs "

```
CREATE TABLE CUSTOMERS(
    ID INT      NOT NULL,
    NAME VARCHAR (20) NOT NULL,
    AGE INT     NOT NULL,
    ADDRESS CHAR (25),
    SALARY DECIMAL (18,2),
    PRIMARY KEY (ID)
);
```

If CUSTOMERS table has already been created, then to add a NOT NULL constraint to the SALARY column in Oracle and MySQL, you would write a query like the one that is shown in the following code block.

**ALTER TABLE CUSTOMERS**

```
MODIFY SALARY DECIMAL (18,2) NOT
NULL;
```

**4. Unique Constraint**

The UNIQUE Constraint prevents two records from having identical values in a column. In the CUSTOMERS table, for example, you might want to prevent two or more people from having an identical age.

**Example**

For example, the following SQL query creates a new table called CUSTOMERS and adds five columns. Here, the AGE column is set to UNIQUE, so that you cannot have two records with the same age.

```
CREATE TABLE CUSTOMERS(
    ID INT      NOT NULL,
    NAME VARCHAR (20) NOT NULL,
    AGE INT     NOT NULL UNIQUE,
    ADDRESS CHAR (25),
    SALARY DECIMAL (18,2),
    PRIMARY KEY (ID)
);
```

If the CUSTOMERS table has already been created, then to add a UNIQUE constraint to the AGE column. You would write a statement like the query that is given in the code block below.

```
ALTER TABLE CUSTOMERS
    MODIFY AGE INT NOT NULL UNIQUE;
```

You can also use the following syntax, which supports naming the constraint in multiple columns as well.

```
ALTER TABLE CUSTOMERS
    ADD CONSTRAINT myUniqueConstraint
    UNIQUE(AGE, SALARY);

DROP a UNIQUE Constraint
```

To drop a UNIQUE constraint, use the following SQL query.

```
ALTER TABLE CUSTOMERS
    DROP CONSTRAINT myUniqueConstraint;
```

If you are using MySQL, then you can use the following syntax "

```
ALTER TABLE CUSTOMERS
    DROP INDEX myUniqueConstraint;
```

## 5. Check constraint

The CHECK Constraint enables a condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and isn't entered the table.

### Example

For example, the following program creates a new table called CUSTOMERS and adds five columns. Here, we add a CHECK with AGE column, so that you cannot have any CUSTOMER who is below 18 years.

```
CREATE TABLE CUSTOMERS
(
    ID INT NOT NULL,
    NAME VARCHAR (20) NOT NULL,
    AGE INT NOT NULL CHECK (AGE >= 18),
    ADDRESS CHAR (25),
    SALARY DECIMAL (18,2),
    PRIMARY KEY (ID)
);
```

If the CUSTOMERS table has already been created, then to add a CHECK constraint to AGE column, you would write a statement like the one given below.

```
ALTER TABLE CUSTOMERS
    MODIFY AGE INT NOT NULL CHECK (AGE >= 18);
```

You can also use the following syntax, which supports naming the constraint in multiple columns as well:

```
ALTER TABLE CUSTOMERS
    ADD CONSTRAINT myCheckConstraint CHECK(AGE >= 18);
    DROP a CHECK Constraint
```

To drop a CHECK constraint, use the following SQL syntax. This syntax does not work with MySQL.

```
ALTER TABLE CUSTOMERS
    DROP CONSTRAINT myCheckConstraint;
```

### 3.2.5 SQL Data types and Schemas

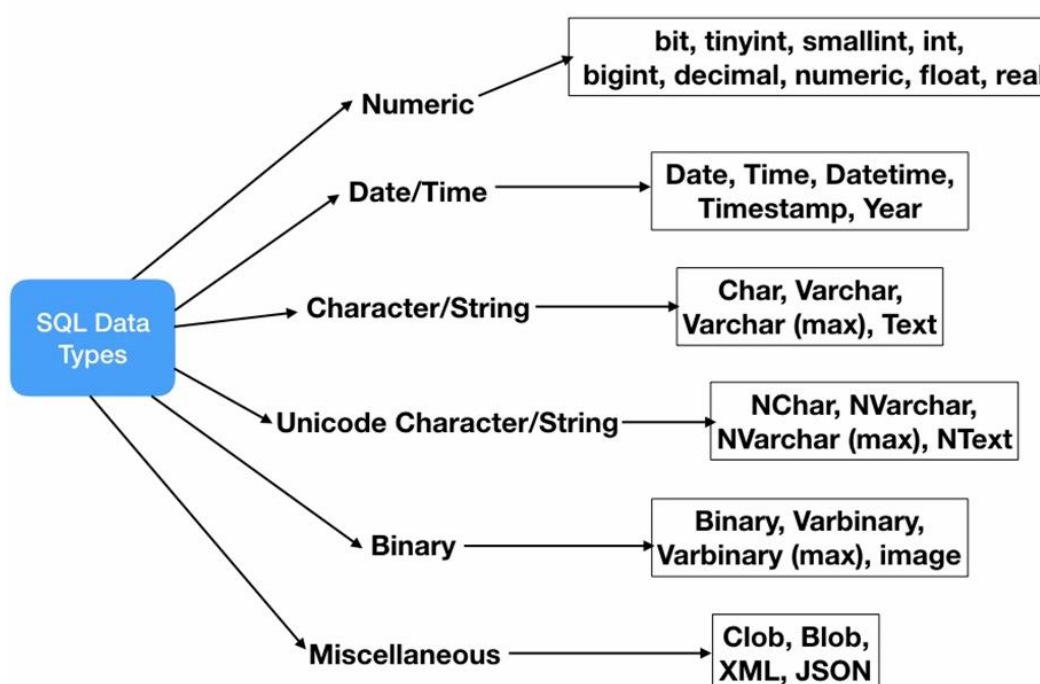
**Q34. List out various types of datatypes supported by SQL.**

*Ans :*

**(Imp.)**

SQL Data Type is an attribute that specifies the type of data of any object. Each column, variable and expression has a related data type in SQL. You can use these data types while creating your tables. You can choose a data type for a table column based on your requirement.

SQL Server offers six categories of data types for your use which are listed below "



#### 1. SQL Numeric Data Types

Datatype	From	To
bit	0	1
tinyint	0	255
smallint	-32,768	32,767
int	-2,147,483,648	2,147,483,647
bigint	-9,223,372,036, 854,775,808	9,223,372,036, 854,775,807
decimal	-10 <sup>38</sup> +1	10 <sup>38</sup> -1
numeric	-10 <sup>38</sup> +1	10 <sup>38</sup> -1
float	-1.79E + 308	1.79E + 308
real	-3.40E + 38	3.40E + 38



**2. SQL Date and Time Data Types**

Datatype	Description
DATE	Stores date in the format YYYY-MM-DD
TIME	Stores time in the format HH:MI:SS
DATETIME	Stores date and time information in the format YYYY-MM-DD HH:MI:SS
TIMESTAMP	Stores number of seconds passed since the Unix epoch ('1970-01-01 00:00:00' UTC)
YEAR	Stores year in 2 digits or 4 digit format. Range 1901 to 2155 in 4-digit format. Range 70 to 69, representing 1970 to 2069.

**3. SQL Character and String Data Types**

Datatype	Description
CHAR	Fixed length with a maximum length of 8,000 characters
VARCHAR	Variable-length storage with a maximum length of 8,000 characters
VARCHAR(max)	Variable-length storage with provided max characters, not supported in MySQL
TEXT	Variable-length storage with maximum size of 2GB data

**Note that all the above data types are for character stream, they should not be used with Unicode data.**

**4. SQL Unicode Character and String Data Types**

Datatype	Description
NCHAR	Fixed length with maximum length of 4,000 characters
NVARCHAR	Variable-length storage with a maximum length of 4,000 characters
NVARCHAR(max)	Variable-length storage with provided max characters
NTEXT	Variable-length storage with a maximum size of 1GB data

**Note that above data types are not supported in MySQL database.**

**5. SQL Binary Data Types**

Datatype	Description
BINARY	Fixed length with a maximum length of 8,000 bytes
VARBINARY	Variable-length storage with a maximum length of 8,000 bytes
VARBINARY(max)	Variable-length storage with provided max bytes
IMAGE	Variable-length storage with maximum size of 2GB binary data

**6. SQL Miscellaneous Data Types**

Datatype	Description
CLOB	Character large objects that can hold up to 2GB
BLOB	For binary large objects
XML	for storing XML data
JSON	for storing JSON data

### 3.2.6 Authorization

#### Q35. Explain authorization process in SQL.

*Ans.:*

- Authorization is finding out if the person, once identified, is permitted to have the resource.
- Authorization explains that what you can do and is handled through the DBMS unless external security procedures are available.
- Database management system allows DBA to give different access rights to the users as per their requirements.
- Basic Authorization we can use any one form or combination of the following basic forms of authorizations
  - i. **Resource authorization:** Authorization to access any system resource. e.g. sharing of database, printer etc.
  - ii. **Alteration Authorization:** Authorization to add attributes or delete attributes from relations
  - iii. **Drop Authorization:** Authorization to drop a relation.

#### Granting of privileges

A system privilege is the right to perform a particular action, or to perform an action on any schema objects of a particular type.

An authorized user may pass on this authorization to other users. This process is called as granting of privileges.

#### Syntax

```
GRANT <privilege list>
ON <relation name or view name>
TO <user/role list>
```

#### Example:

The following grant statement grants user U1,U2 and U3 the select privilege on Emp\_Salary relation:

#### Revoking of privileges:

We can reject the privileges given to particular user with help of revoke statement.

To revoke an authorization, we use the revoke statement.

#### Syntax

```
REVOKE <privilege list>
ON <relation name or view name>
FROM <user/role list> [restrict/cascade]
```

#### Example

The revocation of privileges from user or role may cause other user or roles also have to loose that privileges. This behavior is called cascading of the revoke.

```
Revoke select
ON Emp_Salary
FROM U1,U2,U3.
GRANT select
ON Emp_Salary
TO U1,U2 and U3.
```

### 3.3 ADVANCED SQL

#### 3.3.1 Accessing SQL from a programming language

#### Q36. How can we access SQL from a programming language ?

*Ans.:*

(Imp.)

To access SQL from other programming languages, we can use:

- Dynamic SQL: JDBC and ODBC
- Embedded SQL
- PHP

ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity) serve as APIs for a program to interact with a database server.

In general, the application must make calls to:

1. Connect with the database server
2. Send SQL commands to the database server
3. Fetch tuples of result one-by-one into program variables

**ODBC**

ODBC works with C, C++, C# and Visual Basic (other APIs such as ADO.NET sit on top of ODBC).

ODBC is the standard for application programs communicating with a database server.

The API will:

1. open a connection with a database
2. send queries and updates
3. get back results

ODBC can be used with applications such as GUIs, spreadsheets etc.

**JDBC**

JDBC works with Java.

Along with supporting various features for querying and updating data, and for retrieving query results, JDBC also supports metadata retrieval i.e. retrieving information about the database such as relations present in the database and the names and types of relation attributes.

JDBC connects with the database as follows:

1. open a connection
2. create a "Statement" object
3. execute queries using the Statement object to send queries and fetch results
4. exception mechanism to handle errors

**Embedded SQL**

Embedded SQL refers to embedding SQL queries in another language.

SQL can be embedded in various languages including C, Java and Cobol.

A language into which SQL queries are embedded is referred to as a **host language**, and the SQL structures permitted in the host language comprise **embedded SQL**.

The EXEC SQL statement is used to identify embedded SQL request to the preprocessor:

EXEC SQL <embedded SQL statement> END\_EXEC

Note that this varies by language. Java embedding uses:

# SQL {...};

**PHP**

PHP is a server-side scripting language.

It was mainly developed for the web but can also be used as a general-purpose programming language.

PHP works well with MySQL and can be used in combination with HTML to create a webapp that connects to a database.

**Working**

1. web browser sends HTTP requests and receives HTTP responses
2. PHP script (on the server-side) connects to DBMS and uses query results to produce its output
3. web server calls the PHP script and incorporates its output into the response
4. web browser renders the HTML document from the response

### Executing SQL from PHP

1. connect to server (mysql\_connect)
2. select the database (mysql\_select\_db)
3. run query
4. retrieve row of results (mysql\_fetch\_array)
5. retrieve attributes (foreach)

A typical web-app will have the following components:

#### Login page

Collect credentials and pass them to setup page via POST

##### ➤ Setup page

- Check credentials
- Initialize session and session variables
- Redirect to welcome page

##### ➤ Application pages

Call session\_start(), authenticate the session, and use/ update session variables, as needed

##### ➤ Logout page:

- Calls session\_destroy()
- Redirects to "goodbye" page

### 3.3.2 Functions and procedures

#### Q37. What is Procedure ?

*Ans :*

A procedure is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages. A procedure has a header and a body.

The header consists of the name of the procedure and the parameters or variables passed to the procedure. The body consists of declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.

We can pass parameters to procedures in three ways :

Parameters	Description
IN type	These types of parameters are used to send values to stored procedures.
OUT type	These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.
IN OUT type	These types of parameters are used to send values and get values from stored procedures.

A procedure may or may not return any value.

#### Syntax

```
CREATE [ORREPLACE] PROCEDURE _name
(<Argument> {IN, OUT, INOUT}
<Datatype>,....)
```

IS

Declaration section <variable, constant> ;

BEGIN

Execution section

EXCEPTION

Exception section

END

IS - marks the beginning of the body of the procedure and is similar to DECLARE in anonymous PL/SQL Blocks. The code between IS and BEGIN forms the Declaration section.

The syntax within the brackets [ ] indicate they are optional. By using CREATE OR REPLACE together the procedure is created if no other procedure with the same name exists or the existing procedure is replaced with the current code.

#### Q38. How to execute a procedure ?

*Ans :*

There are two ways to execute a procedure :

- From the SQL prompt : EXECUTE [or EXEC] procedure\_name;

- Within another procedure – simply use the procedure name : procedure\_name;

**Example:**

create table named emp have two column id and salary with number datatype.

```
CREATE OR REPLACE PROCEDURE p1(id IN NUMBER, sal IN NUMBER)
AS
BEGIN
INSERT INTO emp VALUES(id, sal);
    DBMS_OUTPUT.PUT_LINE('VALUE INSERTED.');
```

END;

/

**Output**

Run SQL Command Line	
SQL > set serveroutput on	
SQL > start D://pr.sql	
Procedure created.	
SQL > exec p1(5,4);	
VALUE INSERTED.	
PL/SQL procedure successfully completed.	
SQL > select * from emp;	
ID	SALARY
-----	-----
2	5000

**Q39. What is function ? Write the syntax to declare functions in SQL.**

*Ans :*

A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

**Syntax:**

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
    RETURN return_datatype; {IS, AS}
Declaration_section <variable, constant> ;
BEGIN
Execution_section
Return return_variable;
EXCEPTION
exception_section
Return return_variable;
END;
```

## RETURN TYPE

The header section defines the return type of the function. The return datatype can be any of the oracle datatype like varchar, number etc.

The execution and exception section both should return a value which is of the datatype defined in the header section.

### Q40. How to execute functions?

*Ans :*

A function can be executed in the following ways.

- As a part of a SELECT statement : SELECT emp\_details\_func FROM dual;
- In a PL/SQL Statements like, : dbms\_output.put\_line(emp\_details\_func);

This line displays the value returned by the function.

### Example

```
Create or replace function getsal (no IN number) return number is
sal number(5);
begin
select salary into sal from emp where id=no;
return sal;
end;
/
```

### Output:

Run SQL Command Line

SQL> select \* from emp;

ID	SALARY
----	--------

2	5000
---	------

SQL> start D://fun.sql

Function created.

SQL> select getsal(2) from dual;

GETSAL(2)
-----------

5000
------

In the example we are retrieving the 'salary' of employee with id 2 to variable 'sal'.

The return type of the function is number.

**Q41. What are the differences between functions and procedures in SQL.***Ans :***Function**

Function, in computer programming language context, a set of instructions which takes some input and performs certain tasks. In SQL, a function returns a value.

**Procedure**

Procedure, as well, is a set of instructions which takes input and performs certain task. In SQL, procedure does not return a value. In java, procedure and functions are same and also called sub-routines.

Following are the important differences between SQL Function and SQL Procedure.

Sr. No.	Key	Function	Procedure
1	Definition	A function is used to calculate result using given inputs.	A procedure is used to perform certain task in order.
2	Call	A function can be called by a procedure.	A procedure cannot be called by a function.
3	DML	DML statements cannot be executed within a function.	DML statements can be executed within a procedure.
4	SQL, Query	A function can be called within a query.	A procedure cannot be called within a query.
5	SQL, Call	Whenever a function is called, it is first compiled before being called.	A procedure is compiled once and can be called multiple times without being compiled.
6	SQL, Return	A function returns a value and control to calling function or code.	A procedure returns the control but not any value to calling function or code.
7	try-catch	A function has no support for try-catch	A procedure has support for try-catch blocks.
8	SELECT	A select statement can have a function call.	A select statement can't have a procedure call.
9	Explicit Transaction Handling	A function can not have explicit transaction handling.	A procedure can use explicit transaction handling.

**3.3.3 Triggers****Q42. What is PL/SQL Trigger ? List out the advantages of Triggers ?***Ans :*

Trigger is invoked by Oracle engine automatically whenever a specified event occurs. 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 written to be executed in response to any of the following events.

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

### Advantages of Triggers

These are the following advantages of Triggers:

- Trigger generates some derived column values automatically
- Enforces referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

---

### Q43. Write the syntax to create a trigger.

*Ans :*

#### Syntax for creating trigger

```
CREATE [OR REPLACE ] TRIGGER trigger_name  
{BEFORE | AFTER | INSTEAD OF }  
{INSERT [OR] | UPDATE [OR] | DELETE}  
[OF col_name]  
ON table_name  
[REFERENCING OLD AS o NEW AS n]  
[FOR EACH ROW]  
WHEN (condition)
```

#### **DECLARE**

Declaration-statements

#### **BEGIN**

Executable-statements

#### **EXCEPTION**

Exception-handling-statements

#### **END;**

Here,

- **CREATE [OR REPLACE] TRIGGER trigger\_name:** It creates or replaces an existing trigger with the 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.

### PL/SQL Trigger Example

Let's take a simple example to demonstrate the trigger. In this example, we are using the following CUSTOMERS table:

#### Create table and have records

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	23	Allahabad	20000
2	Suresh	22	Kanpur	22000
3	Mahesh	24	Ghaziabad	24000
4	Chandan	25	Noida	26000
5	Alex	21	Paris	28000
6	Sunita	20	Delhi	30000

#### Create trigger

Let's take a program to create a row level trigger for the CUSTOMERS table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

```

CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.salary - :OLD.salary;
    dbms_output.put_line('Old salary: ' || :OLD.salary);
    dbms_output.put_line('New salary: ' || :NEW.salary);
    dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/

```

After the execution of the above code at SQL Prompt, it produces the following result.

Trigger created.

**Q44. Write a code to get the old salary, new salary and salary difference after the trigger created.**

*Ans :*

**Check the salary difference by procedure**

Use the following code to get the old salary, new salary and salary difference after the trigger created.

**DECLARE**

total\_rows number(2);

**BEGIN**

**UPDATE** customers

**SET** salary = salary + 5000;

**IF** sql%notfound **THEN**

dbms\_output.put\_line('no customers updated');

**ELSIF** sql%found **THEN**

total\_rows := sql%rowcount;

dbms\_output.put\_line('total rows || ' customers updated');

**END IF;**

**END;**

/

**Output:**

Old salary: 20000  
 New salary: 25000  
 Salary difference: 5000  
 Old salary: 22000  
 New salary: 27000  
 Salary difference: 5000  
 Old salary: 24000  
 New salary: 29000  
 Salary difference: 5000  
 Old salary: 26000  
 New salary: 31000  
 Salary difference: 5000  
 Old salary: 28000  
 New salary: 33000  
 Salary difference: 5000  
 Old salary: 30000

New salary: 35000

Salary difference: 5000

6 customers updated

**Note**

As many times you executed this code, the old and new both salary is incremented by 5000 and hence the salary difference is always 5000.

**result**

Old salary: 25000

New salary: 30000

Salary difference: 5000

Old salary: 27000

New salary: 32000

Salary difference: 5000

Old salary: 29000

New salary: 34000

Salary difference: 5000

Old salary: 31000

New salary: 36000

Salary difference: 5000

Old salary: 33000

New salary: 38000

Salary difference: 5000

Old salary: 35000

New salary: 40000

Salary difference: 5000

6 customers updated

**Important Points**

Following are the two very important point and should be noted carefully.

- OLD and NEW references are used for record level triggers these are not available for table level triggers.
- If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.

### 3.3.4 Recursive Queries

**Q45. Explain the process of creating recursive queries.**

*Ans :*

**(Imp.)**

A recursive query is a way to query hierarchies of data, such as an organizational structure, bill-of-materials, and document hierarchy.

Recursion is typically characterized by three steps:

1. Initialization
2. Recursion, or repeated iteration of the logic through the hierarchy
3. Termination

Similarly, a recursive query has three execution phases:

1. Create an initial result set.
2. Recursion based on the existing result set.
3. Final query to return the final result set.

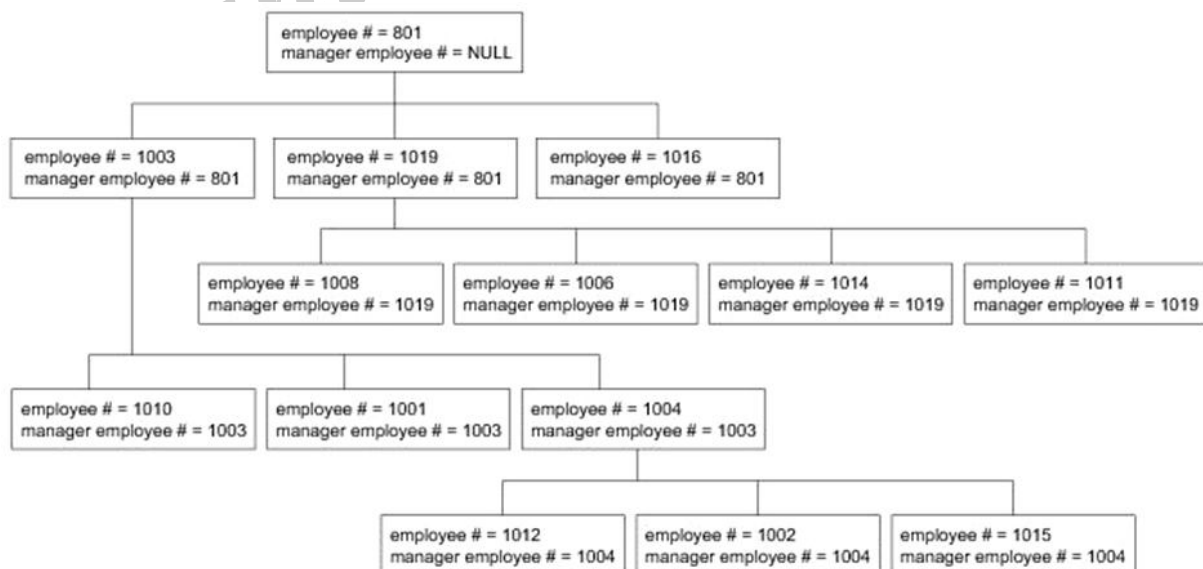
You can specify a recursive query by:

- Preceding a query with the WITH RECURSIVE clause
- Creating a view using the RECURSIVE clause in a CREATE VIEW statement

**Consider the following employee table**

```
CREATE TABLE employee
(employee_number INTEGER
,manager_employee_number INTEGER
,last_name CHAR(20)
,first_name VARCHAR(30));
```

The table represents an organizational structure containing a hierarchy of employee-manager data. The following figure depicts what the employee table looks like hierarchically.



1101A285

The following recursive query retrieves the employee numbers of all employees who directly or indirectly report to the manager with employee\_number 801:

```
WITH RECURSIVE temp_table (employee_
number) AS
( SELECT root.employee_number
FROM employee root
WHERE root.manager_employee_number
= 801
UNION ALL
SELECT indirect.employee_number
FROM temp_table direct, employee indirect
WHERE direct.employee_number =
indirect.manager_employee_number
)
SELECT * FROM temp_table ORDER BY
employee_number;
```

In the example, temp\_table is a temporary named result set that can be referred to in the FROM clause of the recursive statement.

The initial result set is established in temp\_table by the nonrecursive, or seed, statement and contains the employees that report directly to the manager with an employee\_number of 801:

```
SELECT root.employee_number
FROM employee root
WHERE root.manager_employee_number
= 801
```

The recursion takes place by joining each employee in temp\_table with employees who report to the employees in temp\_table. The UNION ALL adds the results to temp\_table.

```
SELECT indirect.employee_number
FROM temp_table direct, employee indirect
WHERE direct.employee_number = indirect.
manager_employee_number.
```

Recursion stops when no new rows are added to temp\_table.

The final query is not part of the recursive WITH clause and extracts the employee information out of temp\_table:

```
SELECT * FROM temp_table ORDER BY
employee_number;
```

Here are the results of the recursive query:

```
employee_number
-----
```

1001

1002

1003

1004

1006

1008

1010

1011

1012

1014

1015

1016

1019

## Short Question and Answers

### 1. What is SQL ?

*Ans :*

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as:

- MS SQL Server using T-SQL,
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc.

IBM developed the original version of SQL, originally called Sequel, as part of the System R project in the early 1970s. The Sequel language has evolved since then, and its name has changed to SQL (Structured Query Language). Many products now support the SQL language. SQL has clearly established itself as the standard relational database language.

In 1986, the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) published an SQL standard, called

SQL-86. ANSI published an extended standard for SQL, SQL-89, in 1989. The next version of the standard was SQL-92 standard, followed by SQL:1999, SQL:2003, SQL:2006, and most recently SQL:2008. The bibliographic notes provide references to these standards.

### 2. What are the advantages of SQL ?

*Ans :*

SQL is widely popular because it offers the following advantages:

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

### 3. What are the differences between grant and revoke

*Ans :*

GRANT	REVOKE
<p><b>GRANT command</b> allows a user to perform certain activities on the database.</p> <p>It grants access privileges for database objects to other users.</p> <p><b>Example:</b></p> <pre>GRANT privilege_name ON object_name TO {     user_name   PUBLIC   role_name } [WITH GRANT OPTION];</pre>	<p><b>REVOKE command</b> disallows a user to perform certain activities.</p> <p>It revokes access privileges for database objects previously granted to other users.</p> <p><b>Example:</b></p> <pre>REVOKE privilege_name ON object_name FROM {     user_name   PUBLIC   role_name }</pre>

### 4. What is the use of Null values?

*Ans :*

The SQL **NULL** is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank.

A field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces.

#### Syntax

The basic syntax of **NULL** while creating a table.

```
SQL> CREATE TABLE CUSTOMERS(
    ID                INT                NOT NULL,
    NAME              VARCHAR (20)      NOT NULL,
    AGE               INT                NOT NULL,
    ADDRESS            CHAR (25),
    SALARY             DECIMAL (18, 2),
    PRIMARY KEY (ID)
);
```

Here, **NOT NULL** signifies that column should always accept an explicit value of the given data type. There are two columns where we did not use NOT NULL, which means these columns could be NULL.

A field with a NULL value is the one that has been left blank during the record creation.

**5. Nested subquery***Ans :*

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow:

- A subquery can be placed in a number of SQL clauses like WHERE clause, FROM clause, HAVING clause.
- You can use Subquery with SELECT, UPDATE, INSERT, DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
- A subquery is a query within another query. The outer query is known as the main query, and the inner query is known as a subquery.
- Subqueries are on the right side of the comparison operator.
- A subquery is enclosed in parentheses.
- In the Subquery, ORDER BY command cannot be used. But GROUP BY command can be used to perform the same function as ORDER BY command.

**6. How can we modify the database using SQL ?***Ans :*

The SQL commands (UPDATE and DELETE) are used to modify the data that is already in the database. The SQL DELETE command uses a WHERE clause.

**SQL UPDATE** statement is used to change the data of the records held by tables. Which rows is to be update, it is decided by a condition. To specify condition, we use WHERE clause.

The UPDATE statement can be written in following form:

```
UPDATE table_name SET [column_name1
= value1,... column_nameN = valueN] [WHERE
condition]
```

**Syntax**

```
UPDATE table_name
SET column_name = expression
WHERE conditions
```

**7. What is Left Join explain with an example.***Ans :***(Imp.)**

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

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

**Syntax**

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

```
SELECT table1.column1, table2.column2...
FROM table1
LEFT JOIN table2
ON table1.common_field
= table2.common_field;
```

Here, the given condition could be any given expression based on your requirement.

**8. What is Right Join?***Ans :*

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

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

**Syntax**

The basic syntax of a **RIGHT JOIN** is as follow.

```
SELECT table1.column1, table2.column2...
FROM table1
RIGHT JOIN table2
ON table1.common_field
= table2.common_field;
```

**9. What is mean by view in SQL? Explain the process of creating, updating and dropping a view ?**

*Ans :*

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

**10. What is transaction? Explain the properties of transaction ?**

*Ans :*

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

A transaction is the propagation of one or more changes to the database. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing a

transaction on that table. It is important to control these transactions to ensure the data integrity and to handle database errors.

Practically, you will club many SQL queries into a group and you will execute all of them together as a part of a transaction.

**Properties**

Transactions have the following four standard properties, usually referred to by the acronym **ACID**.

- **Atomicity:** Ensures that all operations within the work unit are completed successfully. Otherwise, the transaction is aborted at the point of failure and all the previous operations are rolled back to their former state.
- **Consistency:** Ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation:** Enables transactions to operate independently of and transparent to each other.
- **Durability:** Ensures that the result or effect of a committed transaction persists in case of a system failure.

**11. Explain authorization process in SQL.**

*Ans :*

- Authorization is finding out if the person, once identified, is permitted to have the resource.
- Authorization explains that what you can do and is handled through the DBMS unless external security procedures are available.
- Database management system allows DBA to give different access rights to the users as per their requirements.
- Basic Authorization we can use any one form or combination of the following basic forms of authorizations
  - i. **Resource authorization:** Authorization to access any system resource. e.g. sharing of database, printer etc.
  - ii. **Alternation Authorization:** Authorization to add attributes or delete attributes from relations
  - iii. **Drop Authorization:** Authorization to drop a relation.



**12. What is function ? Write the syntax to declare functions in SQL.***Ans :*

A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

**Syntax:**

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
    RETURN return_datatype; {IS, AS}
Declaration_section <variable,constant> ;
BEGIN
Execution_section
Return return_variable;
EXCEPTION
exception section
Return return_variable;
END;
```

**RETURN TYPE**

The header section defines the return type of the function. The return datatype can be any of the oracle datatype like varchar, number etc.

The execution and exception section both should return a value which is of the datatype defined in the header section.

**13. What are the differences between functions and procedures in SQL.***Ans :*

Following are the important differences between SQL Function and SQL Procedure.

Sr. No.	Key	Function	Procedure
1	Definition	A function is used to calculate result using given inputs.	A procedure is used to perform certain task in order.
2	Call	A function can be called by a procedure.	A procedure cannot be called by a function.
3	DML	DML statments cannot be executed within a function.	DML statements can be executed within a procedure.
4	SQL, Query	A function can be called within a query.	A procedure cannot be called within a query.
5	SQL, Call	Whenever a function is called, it is first compiled before being called.	A procedure is compiled once and can be called multiple times without being compiled.
6	SQL, Return	A function returns a value and control to calling function or code.	A procedure returns the control but not any value to calling function or code.
7	try-catch	A function has no support for try-catch	A procedure has support for try-catch blocks.
8	SELECT	A select statement can have a function call.	A select statement can't have a procedure call.
9	Explicit Transaction Handling	A function can not have explicit transaction handling.	A procedure can use explicit transaction handling.

**14. What is PL/SQL Trigger ?***Ans :*

Trigger is invoked by Oracle engine automatically whenever a specified event occurs. 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 written to be executed in response to any of the following events.

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

**Advantages of Triggers**

These are the following advantages of Triggers:

- Trigger generates some derived column values automatically
  - Enforces referential integrity
  - Event logging and storing information on table access
  - Auditing
  - Synchronous replication of tables
  - Imposing security authorizations
  - Preventing invalid transactions
- 

**15. Explain the process of creating recursive queries.***Ans :*

A recursive query is a way to query hierarchies of data, such as an organizational structure, bill-of-materials, and document hierarchy.

Recursion is typically characterized by three steps:

1. Initialization
2. Recursion, or repeated iteration of the logic through the hierarchy
3. Termination

Similarly, a recursive query has three execution phases:

1. Create an initial result set.
2. Recursion based on the existing result set.
3. Final query to return the final result set.

1. An \_\_\_\_\_ is a set of entities of the same type [ a ]
  - a) entity set
  - b) attribute set
  - c) relationset
  - d) entity model
2. The descriptive property of each entity is \_\_\_\_\_ [ b ]
  - a) entity
  - b) attribute
  - c) relation
  - d) model
3. Binary relation ship is \_\_\_\_\_ [ d ]
  - a) 1 : 1
  - b) M : N
  - c) 1 : N
  - d) All
4. Every weak entity can be converted into strong entity using [ c ]
  - a) Generalisation
  - b) Using Aggregation
  - c) Attributes
  - d) None
5. Concepts in which shared subclass take voer relationship from multiple classes is \_\_\_\_\_ [ c ]
  - a) joint inheritance
  - b) partial inheritance
  - c) multiple inheritance
  - d) independent inheritance
6. E-R modeling technique is \_\_\_\_\_ [ a ]
  - a) Top down approach
  - b) Bottom up approach
  - c) left - right approach
  - d) None
7. Relations produced from an ER-model is always in \_\_\_\_\_ [ c ]
  - a) 1 NF
  - b) 2 NF
  - c) 3 NF
  - d) 4 NF
8. A function has partial functional dependencies is \_\_\_\_\_ form [ b ]
  - a) 3 NF
  - b) 2 NF
  - c) 4 NF
  - d) BCNF
9. Consider the relation R(S, T, U, V) and the dependencies  $S \rightarrow T$ ,  $T \rightarrow U$ ,  $U \rightarrow V$  then  $V \rightarrow S$  then the decomposition is in \_\_\_\_\_ form. [ b ]
  - a) Not in 2NF
  - b) 2NF
  - c) 3NF but not in 2NF
  - d) 3NF & 2NF
10. A functional dependency of the form  $X \rightarrow Y$  is trivial if [ b ]
  - a)  $X \subseteq X$
  - b)  $Y \subseteq X$
  - c)  $X \subseteq Y$
  - d)  $X \subset Y$  &  $Y \subset X$

## *Fill in the blanks*

1. An \_\_\_\_\_ is a symmantic way of discribing and defining a business model.
2. ER Model is used to model. The \_\_\_\_\_ of the system (logical view)
3. \_\_\_\_\_ is an object whose inforamtion is stored in DBMS (entity)
4. The \_\_\_\_\_ of a relationship is the number of entitites associated in DBMS (Degree)
5. A \_\_\_\_\_ occurs when 1 to many relationships farout from single entity (fan trap)
6. \_\_\_\_\_ creates a design more accurate to database schema.
7. \_\_\_\_\_ and \_\_\_\_\_ relationship lead to inheritance in DBMS.
8. \_\_\_\_\_ is a process that defines group of entities
9. Generalisation is a \_\_\_\_\_ approach in two level entities.
10. \_\_\_\_\_ anomalies are defined while performing white box testing.
11. \_\_\_\_\_ occurs when two or more independent multi-valued attribute exists
12.  $A \rightarrow B, B \rightarrow C, A \rightarrow C$  is known as \_\_\_\_\_
13. \_\_\_\_\_ is used to organised a data and remove unnecessary duplication of data.
14. BCNF stands for \_\_\_\_\_
15. The \_\_\_\_\_ Datamodel is maintained by documentation ER diagrams and Data dictionary.

### ANSWERS

1. ER Model
2. Logical view
3. Entity
4. Degree
5. Fan trap
6. ERR Model
7. Super class and Subclass
8. Specialisation
9. Button up
10. Dataflow
11. Multivalued dependency
12. Functional dependency
13. Data Redendency
14. Boycee code nornal form
15. Conceptual

## UNIT IV

Transaction Management: Transaction Support–Properties of Transactions, Database Architecture, Concurrency Control–The Need for Concurrency Control, Serializability and Recoverability, Locking Methods, Deadlock, Time Stamping Methods, Multi-version Timestamp Ordering, Optimistic Techniques, Granularity of Data Items, Database Recovery–The Need for Recovery, Transactions and Recovery, Recovery Facilities, Recovery Techniques, Nested Transaction Model. Security: Database Security–Threats, Computer-Based Controls–Authorization, Access Controls, Views, Backup and Recovery, Integrity, Encryption, RAID.

### 4.1 TRANSACTION MANAGEMENT

#### 4.1.1 Transaction Support, Properties of Transactions

**Q1. What is Transaction in DBMS? Explain Properties of Transaction.**

*Ans :*

A transaction is a set of changes that must all be made together. It is a program unit whose execution may or may not change the contents of a database. Transaction is executed as a single unit. If the database was in consistent state before a transaction, then after execution of the transaction also, the database must be in a consistent state. For example, a transfer of money from one bank account to another requires two changes to the database both must succeed or fail together.

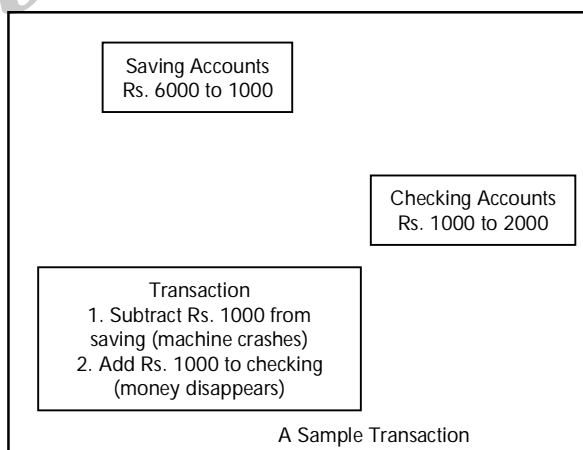
You are working on a system for a bank. A customer goes to the ATM and instructs it to transfer Rs. 1000 from savings to a checking account. This simple transaction requires two steps:

- Subtracting the money from the savings account balance.
- Adding the money to the checking account balance.

The code to create this transaction will require two updates to the database. For example, there will be two SQL statements: one UPDATE command to decrease the balance in savings and a second UPDATE command to increase the balance in the checking account.

You have to consider what would happen if a machine crashed between these two operations. The money has already been subtracted from the savings account will not be added to the checking account. It is lost. You might consider performing the addition to checking first, but then the customer ends up with extra money, and the bank loses.

The point is that both changes must be made successfully. Thus, a transaction is defined as a set of changes that must be made together.

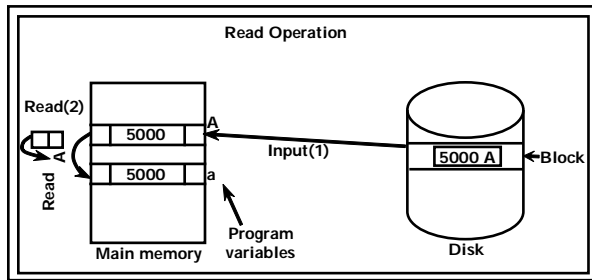


#### Process of Transaction

The transaction is executed as a series of reads and writes of database objects, which are explained below :

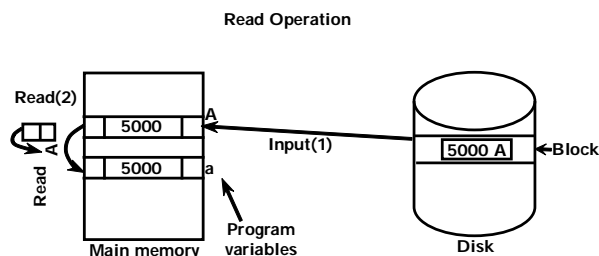
#### Read Operation

To read a database object, it is first brought into main memory from disk, and then its value is copied into a program variable as shown in figure.



### Write Operation

To write a database object, an in-memory copy of the object is first modified and then written to disk.



### Properties

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability “commonly known as ACID properties” in order to ensure accuracy, completeness, and data integrity.

#### 1. Atomicity

This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

#### 2. Consistency

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

#### 3. Durability

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

#### 4. 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.

### Q2. Explain about Transaction States and its Advantages ?

Ans :

There are the following six states in which a transaction may exist :

#### 1. Active

The initial state when the transaction has just started execution.

#### 2. Partially Committed

At any given point of time if the transaction is executing properly, then it is going towards it COMMIT POINT. The values generated during the execution are all stored in volatile storage.

#### 3. Failed

If the transaction fails for some reason. The temporary values are no longer required, and the transaction is set to ROLLBACK. It means that any change made to the database by this transaction up to the point of the failure must be undone. If the failed transaction has withdrawn Rs. 100/- from account A, then the ROLLBACK operation should add Rs 100/- to account A.

**4. Aborted**

When the ROLLBACK operation is over, the database reaches the BFIM. The transaction is now said to have been aborted.

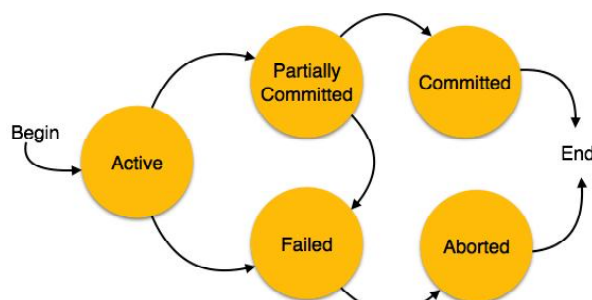
**5. Committed**

If no failure occurs then the transaction reaches the COMMIT POINT. All the temporary values are written to the stable storage and the transaction is said to have been committed.

**6. Terminated**

Either committed or aborted, the transaction finally reaches this state.

The whole process can be described using the following diagram :

**Advantages**

The DBMS interleaves the actions of different transactions to improve performance of system as discussed below :

**(i) Improved Throughput**

Consider that transaction are performed in serial order and active transaction is waiting for a page to be read in from disk, then instead of CPU waiting for a page, it can process another transaction. This is because Input/ Output activity can be done in parallel with the CPU activity. The overlapping of Input/ Output activities of CPU reduces the amount of time disks and processors are idle and increases system throughput (the average number of transaction completed in a given time.)

**(ii) Reduced Waiting time**

Interleaved execution of a short transaction with a long transaction usually allows the short

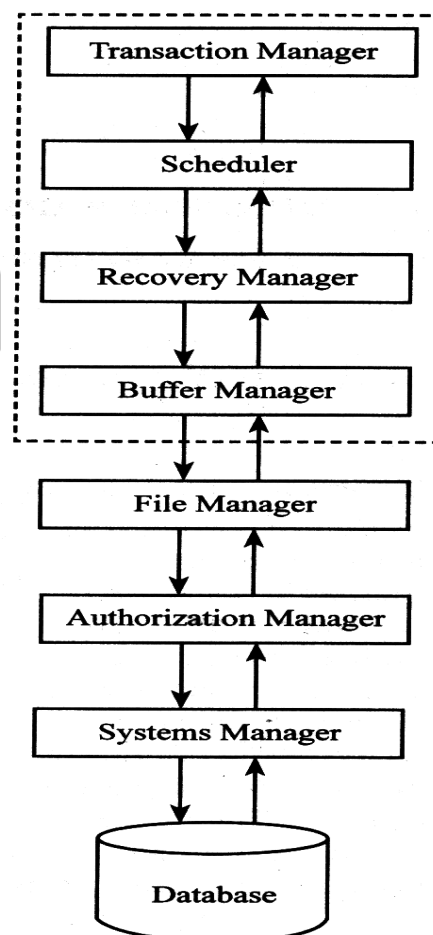
transaction to complete quickly. In serial execution a short transaction could get stuck behind a long transaction leading to unpredictable delays in response time or average time taken to complete a transaction.

**4.1.2 Database Architecture****Q3. State the architecture of transaction management.**

*Ans :*

**(Imp.)**

Typical database architecture with modules that manage transactions, concurrency control and recovery is shown in the figure below:

**1. Transaction Manager**

It manages the transactions so as to ensure that data remain in consistent state even after the system failures. It also enables the execution of concurrent transactions without any conflicts.

**2. Scheduler**

The task of scheduler is to apply appropriate concurrency control mechanism while interacting with the transaction manager. In case if the adopted mechanism is based on the concept of locking it acts as a lock manager. Use of scheduler increases the rate of concurrent transactions while eliminating the risk of interfering each other.

**3. Recovery Manager**

Recovery manager is responsible for restoring the failed database to a consistent state. It typically roll back the failed transaction.

**4. Buffer Manager**

It handles the transfer of data from disk onto the main memory and decides what data must be kept in main memory.

**5. File Manager**

It manages the process of allocating disk and data structures that are used for representing the information saved on disk.

**6. Authorization Manager**

It checks the authority of users and allows only authorized users to access the data.

**7. System Manager**

System manager acts as a query processor which retrieves the data from database.

**4.2 CONCURRENCY CONTROL****4.2.1 The Need for Concurrency Control**

**Q4. What is the purpose of concurrency control for a database?**

(OR)

**Why do we need Concurrency control?**

(OR)

**Why DBMS needs a concurrency control?**

*Ans :*

(Imp.)

1. Transaction management (TM) handles all transactions properly in DBMS. Database transactions are the events or activities such as series of data read/write operations on data object(s) stored in database system.
2. Concurrency control (CC) is a process to ensure that data is updated correctly and appropriately when multiple transactions are concurrently executed in DBMS.

**Needs**

In general, concurrency control is an essential part of TM. It is a mechanism for correctness when two or more database transactions that access the same data or data set are executed concurrently with time overlap. According to Wikipedia.org, if multiple transactions are executed serially or sequentially, data is consistent in a database. However, if concurrent transactions with interleaving operations are executed, some unexpected data and inconsistent result may occur. Data interference is usually caused by a write operation among transactions on the same set of data in DBMS.

There are two main kinds of concurrency control mechanisms :

- **Pessimistic (conservative) concurrency control:** The pessimistic concurrency control delays the transactions if they conflict with other transactions at some time in the future by locking or a time-stamping technique.
- **Optimistic concurrency control:** According to Kung and Robinson (1981), the optimistic concurrency control, that assumes that the conflict is rare, allows concurrent transactions to proceed without imposing delays to ensure serializability then check conflict only at the end, when a transaction commits. Notice that there is another



mechanism, semi-optimistic technique, which uses lock operations in some situations (if they may violate some rules), and does not lock in other circumstances.

- **The pros and cons of the pessimistic and optimistic concurrency control mechanisms:** Both pessimistic and optimistic concurrency control mechanisms provide different performance, e.g., the different average transaction completion rates or throughput, depending on transaction types mix, computing level of parallelism, and other events.

According to Vallejo, Sanyal, Harris, Vallejo, Beivide, Unsal, Valero (2011), there is a tradeoff between the concurrency control techniques. Their pros and cons are shown below:

For pessimistic concurrency control, the strength is :

- Guarantee that all transactions can be executed correctly.
- Data is properly consistent by either rolling back to the previous state (Abort operation) or new content (Commit operation) when the transaction conflict is cleared.
- Database is relatively stable and reliable.

**Its weakness is :**

- Transactions are slow due to the delay by locking or time-stamping event.
- Runtime is longer. Transaction latency increases significantly.
- Throughput or the amount of work (e.g. read/write, update, rollback operations, etc.) is reduced.

For optimistic concurrency control, the strength is:

- Transactions are executed more efficiently.
- Data content is relatively safe.
- Throughput is much higher.

**Its weakness is :**

- There is a risk of data interference among concurrent transactions since it transactions conflict may occur during execution. In this case, data is no longer correct.
- Database may have some hidden errors with inconsistent data; even conflict check is performed at the end of transactions.
- Transactions may be in deadlock that causes the system to hang.

Many users often encounter the data interference issue in database management system in stock markets. The simple example addresses the inconsistent data issue as shown below.

Multiple analysts or investors from Fidelity Investments, LLC access a client's fund in a database for stock trading from either Dow, Bonds or Mutual Funds. Both analysts A, B perform some transactions to transfer some amount of money to buy stocks from different funds for their daily works. In this scenario, the nested transaction is used in database. At scheduled step T7, a transaction manager faces inconsistent data (i.e. \$800 or \$1000) in the client's balance. The Operating System can execute either thread A for a balanced result of 1000 or thread B for a balanced result of 800. Either case the balance content is incorrect. For pessimistic technique, the transaction B may be delayed for a long time or in a deadlock. For optimistic technique, it goes through with incorrect data in the database. Furthermore, the micro execution of the software threads is unknown, ambiguous and out of control. Database is in the unknown state.

Steps	Operations	Analyst A's Transactions	Analyst B's Transactions	Clients' Balance Bal \$ US
T1	Beginning	Start		1000
T2	A reads balance	Read(Bal)		1000
T3	Transfer 100 to Fund A	Bal = Bal - 100	Start	1000
T4	B reads balance		Read(Bal)	1000
T5	A updates balance Transfer 200 to Fund B	Write (Bal)	Bal = Bal - 200	900
T6	B updates balance		Write(Bal)	800
T7	A aborts, B commits	Rollback	Commit	1000? or 800?
T8	End	End	Edn	Conflict

In conclusion, concurrency control is one of the primary mechanisms in transaction management to provide integrity of data and safety in DBMS. Today, with hundred thousand or more transactions in a few minutes, transaction management and concurrency control become much more complex and sophisticated. Two pessimistic and optimistic mechanisms are still popular, but other techniques such as semi-optimistic are also applied in DBMS for higher performance, better throughput, more accurate results, and faster run time.

**Q5. What are the difference concurrent control problems? Explain with examples.**

*Ans :*

The coordination of the simultaneous execution of transactions in a multiuser database system is known as concurrency control. The objective of concurrency control is to ensure the serializability of transactions in a multiuser database environment. Concurrency control is important because the simultaneous execution of transactions over a shared database can create several data integrity and consistency problems. The three main problems are lost updates, uncommitted data, and inconsistent retrievals.

#### Lost Updates

The lost update problem occurs when two concurrent transactions, T1 and T2, are updating the same data element and one of the updates is lost (overwritten by the other transaction). Consider the following PRODUCT table example.

One of the PRODUCT table's attributes is a product's quantity on hand (PROD\_QOH).

Assume that you have a product whose current PROD\_QOH value is 35. Also assume that two concurrent transactions, T1 and T2, occur that update the PROD\_QOH value for some item in the PRODUCT table.

The transactions are as follows.

Two concurrent transactions update PROD\_QOH :

#### Transaction Operation

T1: Purchase 100 units       $\text{PROD\_QOH} = \text{PROD\_QOH} + 100$

T2: Sell 30 units       $\text{PROD\_QOH} = \text{PROD\_QOH} - 30$

The Following table shows the serial execution of those transactions under normal circumstances, yielding the correct answer  $\text{PROD\_QOH} = 105$ .

Time	Transaction	Step	Stored Value
1	T1	Read PROD_QOH	35
2	T1	$\text{PROD\_QOH} = 35 + 100$	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH	135
5	T2	$\text{PROD\_QOH} = 135 - 30$	
6	T2	Write PROD_QOH	105

But suppose that a transaction is able to read a product's PROD\_QOH value from the table before a previous transaction (using the same product) has been committed.

The sequence depicted in the following Table shows how the lost update problem can arise.

Note that the first transaction (T1) has not yet been committed when the second transaction (T2) is executed. Therefore, T2 still operates on the value 35, and its subtraction yields 5 in memory. In the meantime, T1 writes the value 135 to disk, which is promptly overwritten by T2. In short, the addition of 100 units is "lost" during the process.

Time	Transaction	Step	Stored Value
1	T1	Read PROD_QOH	35
2	T1	Read PROD_QOH	35
3	T1	$\text{PROD\_QOH} = 35 + 100$	
4	T2	$\text{PROD\_QOH} = 35 - 30$	
5	T1	Write PRPD_QOH (Lost update)	135
6	T2	Write PROD_QOH	5

### Uncommitted Data

The phenomenon of uncommitted data occurs when two transactions, T1 and T2, are executed concurrently and the first transaction (T1) is rolled back after the second transaction (T2) has already accessed the uncommitted data—thus violating the isolation property of transactions.

To illustrate that possibility, let's use the same transactions described during the lost updates discussion. T1 has two atomic parts to it, one of which is the update of the inventory, the other possibly being the update of the invoice total (not shown).

T1 is forced to roll back due to an error during the updating of the invoice's total; hence, it rolls back all the way, undoing the inventory update as well. This time, the T1 transaction is rolled back to eliminate the addition of the 100 units. Because T2 subtracts 30 from the original 35 units, the correct answer should be 5.

### Transaction Operation

T1: Purchase 100 units      $\text{PROD\_QOH} = \text{PROD\_QOH} + 100$  (Rolled back)

T2: Sell 30 units              $\text{PROD\_QOH} = \text{PROD\_QOH} - 30$

The following Table shows how, under normal circumstances, the serial execution of those transactions yields the correct answer.

Time	Transaction	Step	Stored Value
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T1	*****ROLLBACK*****	35
5	T2	Read PROD_QOH	35
6	T2	PROD_QOH = 35 - 30	
7	T2	Write PROD_QOH	5

The following Table shows how the uncommitted data problem can arise when the ROLLBACK is completed after T2 has begun its execution.

Time	Transaction	Step	Stored Value
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH (Read uncommitted data) →	135
5	T2	PROD_QOH = 135 - 30	
6	T1	**** ROLLBACK ****	35
7	T2	Write PROD_QOH	5

### Inconsistent Retrievals

Inconsistent retrievals occur when a transaction accesses data before and after another transaction(s) finish working with such data. For example, an inconsistent retrieval would occur if transaction T1 calculated some summary (aggregate) function over a set of data while another transaction (T2) was updating the same data. The problem is that the transaction might read some data before they are changed and other data after they are changed, thereby yielding inconsistent results.

To illustrate that problem, assume the following conditions :

- T1 calculates the total quantity on hand of the products stored in the PRODUCT table.
- At the same time, T2 updates the quantity on hand (PROD\_QOH) for two of the PRODUCT table's products.

The two transactions are shown in the following Table :

TRANSACTION T1		TRANSACTION T2	
SELECT	SUM(PROD_QOH)	UPDATE	PRODUCT
FROM	PRODUCT	SET	PROD_QOH = PROD_QOH + 10
			WHERE PROD_CODE = '1546-QQ2'
		UPDATE	PRODUCT
		SET	PROD_QOH = PROD_QOH - 10
		WHERE	PROD_CODE = '1158-QW1'
		COMMIT;	

While T1 calculates the total quantity on hand (PROD\_QOH) for all items, T2 represents the correction of a typing error: the user added 10 units to product 1558-QW1's PROD\_QOH but meant to add the 10 units to product 1546-QQ2's PROD\_QOH. To correct the problem, the user adds 10 to product 1546-QQ2's PROD\_QOH and subtracts 10 from product 1558-QW1's PROD\_QOH. The initial and final PROD\_QOH values are reflected in the following Table

	BEFORE	AFTER
PROD_CODE	PROD_QOH	PROD_QOH
11QER/31	8	8
13-Q2/P2	32	32
1546-QQ2	15	(15 + 10) → 25
1558-QW1	23	(23 - 10) → 13
2232-QTY	8	8
2232-QWE	6	6
<b>Total</b>	<b>92</b>	<b>92</b>

The following table demonstrates that inconsistent retrievals are possible during the transaction execution, making the result of T1's execution incorrect. The "After" summation shown in Table reflects the fact that the value of 25 for product 1546-QQ2 was read after the WRITE statement was completed. Therefore, the "After" total is  $40 + 25 = 65$ . The "Before" total reflects the fact that the value of 23 for product 1558-QW1 was read before the next WRITE statement was completed to reflect the corrected update of 13. Therefore, the "Before" total is  $65 + 23 = 88$ .

TIME	TRANSACTION	ACTION	VALUE	TOTAL
1	T1	Read PROD_QOH for PROD_CODE = '11QER/31'	8	8
2	T1	Read PROD_QOH for PROD_CODE = '13-Q2/P2'	32	40
3	T2	Read PROD_QOH for PROD_CODE = '1546-QQ2'	15	
4	T2	PROD_QOH = 15 + 10		
5	T2	Write PROD_QOH for PROD_CODE = '1546-QQ2'	25	
6	T1	Read PROD_QOH for PROD_CODE = '1546-QQ2'	25	(After) 65
7	T1	Read PROD_QOH for PROD_CODE = '1558-QW1'	23	(Before) 88
8	T2	Read PROD_QOH for PROD_CODE = '1558-QW1'	23	
9	T2	PROD_QOH = 23 - 10		
10	T2	Write PROD_QOH for PROD_CODE = '1558-QW1'	13	
11	T2	***** COMMIT *****		
12	T1	Read PROD_QOH for PROD_CODE = '2232-QTY'	8	96
13	T1	Read PROD_QOH for PROD_CODE = '2232-QWE'	6	102

The computed answer of 102 is obviously wrong because you know from the previous Table that the correct answer is 92. Unless the DBMS exercises concurrency control, a multiuser database environment can create havoc within the information system.

### 4.3 SERIALIZABILITY

**Q6. Define Serializability. State the importance of Serializability?**

*Ans :*

(Imp.)

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

- **Schedule:** A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
- **Serial Schedule:** It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

In a multi-transaction environment, serial schedules are considered as a benchmark. The execution sequence of an instruction in a transaction cannot be changed, but two transactions can have their instructions executed in a random fashion. This execution does no harm if two transactions are mutually independent and working on different segments of data; but in case these two transactions are working on the same data, then the results may vary. This ever-varying result may bring the database to an inconsistent state.

To resolve this problem, we allow parallel execution of a transaction schedule, if its transactions are either Serializable or have some equivalence relation among them.

### Equivalence Schedules

An equivalence schedule can be of the following types:

1. **Result Equivalence:** If two schedules produce the same result after execution, they are said to be result equivalent. They may yield the same result for some value and different results for another set of values. That's why this equivalence is not generally considered significant.
2. **View Equivalence:** Two schedules would be view equivalence if the transactions in both the schedules perform similar actions in a similar manner.

### For example

- If T reads the initial data in S1, then it also reads the initial data in S2.
- If T reads the value written by J in S1, then it also reads the value written by J in S2.
- If T performs the final write on the data value in S1, then it also performs the final write on the data value in S2.

### Conflict Equivalence

Two schedules would be conflicting if they have the following properties -

- Both belong to separate transactions.
- Both accesses the same data item.
- At least one of them is "write" operation.

Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if

- Both the schedules contain the same set of Transactions.
- The order of conflicting pairs of operation is maintained in both the schedules.

**Note:** View equivalent schedules are view serializable and conflict equivalent schedules are conflict serializable. All conflict serializable schedules are view serializable too.

**Q7. Explain about Conflict Serializability?***Ans :*

As discussed in Concurrency control, serial schedules have less resource utilization and low throughput. To improve it, two or more transactions are run concurrently. But concurrency of transactions may lead to inconsistency in database. To avoid this, we need to check whether these concurrent schedules are Serializable or not.

- **Conflict Serializable:** A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.
- **Conflicting operations:** Two operations are said to be conflicting if all conditions satisfy:
  - They belong to different transaction
  - They operation on same data item
  - At Least one of them is a write operation

**Example :**

- **Conflicting** operations pair  $(R_1(A), W_2(A))$  because they belong to two different transactions on same data item A and one of them is write operation.
- Similarly,  $(W_1(A), W_2(A))$  and  $(W_1(A), R_2(A))$  pairs are also conflicting.
- On the other hand,  $(R_1(A), W_2(B))$  pair is non-conflicting because they operate on different data item.
- Similarly,  $(W_1(A), W_2(B))$  pair is non-conflicting.

**Consider the following schedule :**

S1:  $R_1(A), W_1(A), R_2(A), W_2(A), R_1(B), W_1(B), R_2(B), W_2(B)$

If  $O_i$  and  $O_j$  are two operations in a transaction and  $O_i < O_j$  ( $O_i$  is executed before  $O_j$ ), same order will follow in schedule as well. Using this property, we can get two transactions of schedule S1 as:

T1:  $R_1(A), W_1(A), R_1(B), W_1(B)$

T2:  $R_2(A), W_2(A), R_2(B), W_2(B)$

**Possible Serial Schedules are: T1->T2 or T2->T1**

- Swapping non-conflicting operations  $R_2(A)$  and  $R_1(B)$  in S1, the schedule becomes,

**S11:  $R_1(A), W_1(A), R_1(B), W_2(A), R_2(A), W_1(B), R_2(B), W_2(B)$**

- Similarly, swapping non-conflicting operations  $W_2(A)$  and  $W_1(B)$  in S11, the schedule becomes,

**S12:  $R_1(A), W_1(A), R_1(B), W_1(B), R_2(A), W_2(A), R_2(B), W_2(B)$**

S12 is a serial schedule in which all operations of T1 are performed before starting any operation of T2. Since S has been transformed into a serial schedule S12 by swapping non-conflicting operations of S1, S1 is conflict serializable.

Let us take another Schedule

**S2:  $R_2(A), W_2(A), R_1(A), W_1(A), R_1(B), W_1(B), R_2(B), W_2(B)$**

Two transactions will be

T1:  $R_1(A), W_1(A), R_1(B), W_1(B)$

T2:  $R_2(A), W_2(A), R_2(B), W_2(B)$

**Possible Serial Schedules are : T1→T2 or T2→T1**

Original Schedule is:

**S2 : R<sub>2</sub>(A), W<sub>2</sub>(A), R<sub>1</sub>(A), W<sub>1</sub>(A), R<sub>1</sub>(B), W<sub>1</sub>(B), R<sub>2</sub>(B), W<sub>2</sub>(B)**

Swapping non-conflicting operations R<sub>1</sub>(A) and R<sub>2</sub>(B) in S2, the schedule becomes,

**S21: R<sub>2</sub>(A), W<sub>2</sub>(A), R<sub>2</sub>(B), W<sub>1</sub>(A), R<sub>1</sub>(B), W<sub>1</sub>(B), R<sub>1</sub>(A), W<sub>2</sub>(B)**

Similarly, swapping non-conflicting operations W<sub>1</sub>(A) and W<sub>2</sub>(B) in S21, the schedule becomes,

**S22 : R<sub>2</sub>(A), W<sub>2</sub>(A), R<sub>2</sub>(B), W<sub>2</sub>(B), R<sub>1</sub>(B), W<sub>1</sub>(B), R<sub>1</sub>(A), W<sub>1</sub>(A)**

In schedule S22, all operations of T2 are performed first, but operations of T1 are not in order (order should be R<sub>1</sub>(A), W<sub>1</sub>(A), R<sub>1</sub>(B), W<sub>1</sub>(B)). So S2 is not conflict Serializable.

### Conflict Equivalent

Two schedules are said to be conflict equivalent when one can be transformed to another by swapping non-conflicting operations. In the example discussed above, S11 is conflict equivalent to S1 (S1 can be converted to S11 by swapping non-conflicting operations). Similarly, S11 is conflict equivalent to S12 and so on.

#### Note 1:

Although S2 is not conflict serializable, but still it is conflict equivalent to S21 and S21 because S2 can be converted to S21 and S22 by swapping non-conflicting operations.

#### Note 2:

The schedule which is conflict serializable is always conflict equivalent to one of the serial schedule. S1 schedule discussed above (which is conflict serializable) is equivalent to serial schedule (T1→T2).

<b>R1(A);W2(A)</b>	It is a conflict pair, Because T1 is reading initial value of A and T2 writing a different value for A and if we interchange their order then T1 will read the value of A written by T2, thus behavior will change.
<b>W1(A);R2(A)</b> <b>W1(A); W2(A)</b>	Conflict Pairs
<b>R1(A); R2(A)</b>	Not a Conflict Pair
<b>R1(A);W2(B)</b> <b>W1(A); R2(B)</b>	Not a Conflict Pairs because they are acting on two different values, A and B.
<b>R1(A); W1(A)</b>	We can't interchange order within same transaction. Therefore, No conflict

### Testing for Conflict Serializability

#### Method 1:

- First write the given schedule in a linear way.
- Find the conflict pairs (RW, WR, WW) on same variable by different transactions.
- Whenever conflict pairs are find, write the dependency relation like T<sub>i</sub> → T<sub>j</sub>, if conflict pair is from T<sub>i</sub> to T<sub>j</sub>. For example, (W1(A), R2(A)) ⇒ T1 → T2
- Check to see if there is a cycle formed,



- If yes= not conflict serializable
- No= we get a sequence and hence are conflict serializable.

**Method 2 :**

To test the conflict serializability, we can draw a Graph  $G = (V, E)$  where

$V$  = Vertices = number of transactions

$E$  = Edges = for conflicting pair

1. Create node for each transaction.
2. Find the conflict pairs (RW, WR, WW) on same variable by different transactions.
3. Draw edge from the schedule for each conflict pair such that for example,  $W_2(B)$ ,  $R_1(A)$  is conflict pair, draw edge from T2 to T1 i.e. T2 must be executed before T1.
4. Testing conditions for conflict serializability of schedule
  - a. If precedence graph is cyclic non conflict serializable schedule
  - b. If precedence graph is a acyclic conflict serializable schedule

**4.4 RECOVERABILITY**
**Q8. Discuss in detail about the Recoverability of Schedules ?**

*Ans :*

(Imp.)

A transaction may not execute completely due to hardware failure, system crash or software issues. In that case, we have to rollback the failed transaction. But some other transaction may also have used values produced by failed transaction. So we have to rollback those transactions as well.

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-1000;	A=4000			A=5000
W(A);	A=4000			A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
		Commit;		
Failure point				
Commit;				

Above table shows a schedule with two transactions, T1 reads and writes A and that value is read and written by T2. T2 commits. But later on, T1 fails. So we have to rollback T1. Since T2 has read the value written by T1, it should also be rolled back. But we have already committed that. So this schedule is irrecoverable schedule.

**Irrecoverable Schedule :** When  $T_j$  is reading the value updated by  $T_i$  and  $T_j$  is committed before commit of  $T_i$ , the schedule will be irrecoverable.

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-1000;	A=4000			A=5000
W(A);	A=4000			A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
Failure				
Commit;				
		Commit;		

Table shows a schedule with two transactions, T1 reads and writes A and that value is read and written by T2. But later on, T1 fails. So we have to rollback T1. Since T2 has read the value written by T1, it should also be roll backed. As it has not committed, we can rollback T2 as well. So it is recoverable with cascading rollback.

Recoverable with cascading rollback: If Tj is reading value updated by Ti and commit of Tj is delayed till commit of Ti, the schedule is called recoverable with cascading rollback.

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-1000;	A=4000			A=5000
W(A);	A=4000			A=4000
Commit;				A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
		Commit;		

Table shows a schedule with two transactions, T1 reads and writes A and commits and that value is read by T2. But if T1 fails before commit, no other transaction has read its value, so there is no need to rollback other transaction. So this is a cascadeless recoverable schedule.

#### 4.5 LOCKING METHODS

**Q9. What is Lock? Explain various type of Locks in DBMS.**

*Ans :*

**(Imp.)**

Transaction processing systems usually allow multiple transactions to run concurrently. By allowing multiple transactions to run concurrently will improve the performance of the system in terms of increased throughput or improved response time, but this allows causes several complications with consistency of the data. Ensuring consistency in spite of concurrent execution of transaction require extra work, which is performed by the concurrency controller system of DBMS.

A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it. Generally, there is one lock for each data item in the database. Locks are used as a means of synchronizing the access by concurrent transactions to the database item.

### Types of Locks

Several types of locks are used in concurrency control. To introduce locking concepts gradually, we first discuss binary locks, which are simple but restrictive and so are not used in practice. We then discuss shared/exclusive locks, which provide more general locking capabilities and are used in practical database locking schemes.

#### 1. Binary Locks

A binary lock can have two states or values: locked and unlocked.

A distinct lock is associated with each database item  $A$ . If the value of the lock on  $A$  is 1, item  $A$  cannot be accessed by a database operation that requests the item. If the value of the lock on  $A$  is 0 then item can be accessed when requested. We refer to the current value of the lock associated with item  $A$  as  $LOCK(A)$ . There are two operations, lock item and unlock item are used with binary locking. A transaction requests access to an item  $A$  by first issuing a lockitem( $A$ ) operation. If  $LOCK(A) = 1$ , the transaction is forced to wait. If  $LOCK(A) = 0$  it is set to 1 (the transaction locks the item) and the transaction is allowed to access item  $A$ . When the transaction is through using the item, it issues an unlock item( $A$ ) operation, which sets  $LOCK(A)$  to 0 (unlocks the item) so that  $A$  may be accessed by other transactions. Hence binary lock enforces mutual exclusion on the data item.

#### Rules of Binary Locks

If the simple binary locking scheme described here is used, every transaction must obey the following rules :

1. A transaction must issue the operation lock\_item( $A$ ) before any read\_item( $A$ ) or write\_item operations are performed in  $T$ .
2. A transaction  $T$  must issue the operation unlock\_item( $A$ ) after all read\_item( $A$ ) and
3. write\_item( $A$ ) operations are completed in  $T$ .
4. A transaction  $T$  will not issue a lock\_item( $A$ ) operation if it already holds the lock on Item  $A$ .
5. A transaction  $T$  will not issue an unlock\_item( $A$ ) operation unless it already holds the lock on item  $A$ .
6. The lock manager module of the DBMS can enforce these rules. Between the Lock\_item( $A$ ) and unlock\_item( $A$ ) operations in transaction  $T$ , is said to hold the lock on item  $A$ . At most one transaction can hold the lock on a particular item. Thus no two transactions can access the same item concurrently.

#### Disadvantages of Binary Locks

##### i) Share/Exclusive (for Read/Write) Locks

We should allow several transactions to access the same item  $A$  if they all access  $A$  for reading purposes only. However, if a transaction is to write an item  $A$ , it must have exclusive access to  $A$ . For this purpose, a different type of lock called a multiple-mode lock is used. In this scheme there are shared/exclusive or read/write locks are used.

##### ii) Locking Operations

There are three locking operations called read\_lock( $A$ ), write\_lock( $A$ ) and unlock( $A$ ) represented as lock-S( $A$ ), lock-X( $A$ ), unlock( $A$ ) (Here, S indicates shared lock, X indicates exclusive lock) can be performed on a data item. A lock associated with an item  $A$ ,  $LOCK(A)$ , now has three possible states: "read-locked", "write-locked," or "unlocked." A read-locked item is also called share-locked item because other transactions are allowed to read the item, whereas a write-locked item is caused exclusive-locked, because a single transaction exclusively holds the lock on the item.

## 2. Compatibility of Locks

Suppose that there are A and B two different locking modes. If a transaction T1 requests a lock of mode on item Q on which transaction T2 currently hold a lock of mode B. If transaction can be granted lock, in spite of the presence of the mode B lock, then we say mode A is compatible with mode B. Such a function is shown in one matrix as shown below :

	S	X
S	true	false
X	false	false

**Compatibility Graph**

The graphs shows that if two transactions only read the same data object they do not conflict, but if one transaction writes a data object and another either read or write the same data object, then they conflict with each other. A transaction requests a shared lock on data item Q by executing the lock-S(Q) instruction. Similarly, an exclusive lock is requested through the lock- X(Q) instruction. A data item Q can be unlocked via the unlock(Q) instruction.

To access a data item, transaction T1 must first lock that item. If the data item is already locked by another transaction in an incompatible mode, the concurrency control manager will not grant the lock until all incompatible locks held by other transactions have been released. Thus, T1 is made to wait until all incompatible locks held by other transactions have been released.

## 4.6 DEADLOCK PREVENTION AND DETECTION

**Q10. Define deadlock. State the prevention and detection of deadlock?**

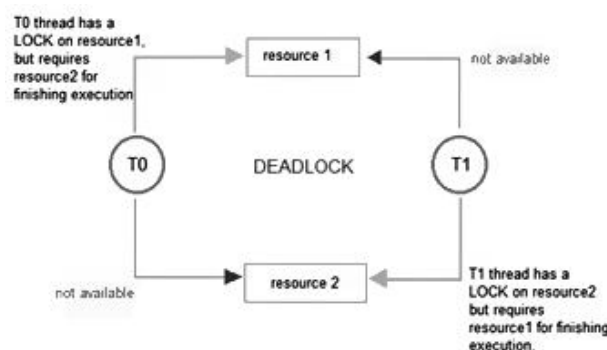
*Ans :*

**(Imp.)**

In a multi-process system, deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.

For example, assume a set of transactions  $\{T_0, T_1, T_2, \dots, T_n\}$ .  $T_0$  needs a resource X to complete its task. Resource X is held by  $T_1$ , and  $T_1$  is waiting for a resource Y, which is held by  $T_2$ .  $T_2$  is waiting for resource Z, which is held by  $T_0$ . Thus, all the processes wait for each other to release resources. In this situation, none of the processes can finish their task. This situation is known as a deadlock.

Deadlocks are not healthy for a system. In case a system is stuck in a deadlock, the transactions involved in the deadlock are either rolled back or restarted.

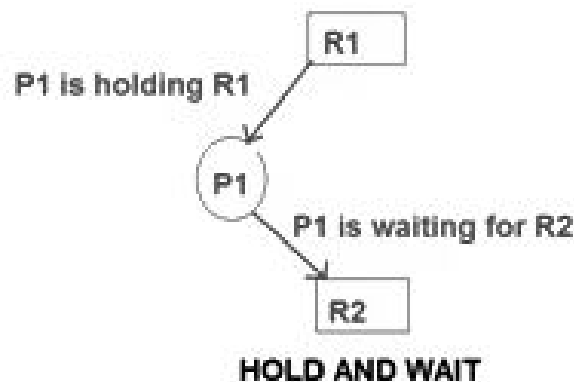


### Deadlock detection

Resource scheduler is one that keeps the track of resources allocated to and requested by processes. Thus, if there is a deadlock it is known to the resource scheduler. This is how a deadlock is detected.

Once a deadlock is detected it is being corrected by following methods :

- **Terminating processes involved in deadlock** : Terminating all the processes involved in deadlock or terminating process one by one until deadlock is resolved can be the solutions but both of these approaches are not good. Terminating all processes cost high and partial work done by processes gets lost. Terminating one by one takes lot of time because each time a process is terminated, it needs to check whether the deadlock is resolved or not. Thus, the best approach is considering process age and priority while terminating them during a deadlock condition.
- **Resource Preemption**: Another approach can be the preemption of resources and allocation of them to the other processes until the deadlock is resolved.



### Deadlock Prevention

Deadlock Prevention ensures that the system never enters a deadlock state.

Following are the requirements to free the deadlock :

- **No Mutual Exclusion** : No Mutual Exclusion means removing all the resources that are sharable.
- **No Hold and Wait** : Removing hold and wait condition can be done if a process acquires all the resources that are needed before starting out.
- **Allow Preemption** : Allowing preemption is as good as removing mutual exclusion. The only need is to restore the state of the resource for the preempted process rather than letting it in at the same time as the preemptor.
- **Removing Circular Wait** : The circular wait can be removed only if the resources are maintained in a hierarchy and process can hold the resources in increasing the order of precedence.

### Deadlock Avoidance

Aborting a transaction is not always a practical approach. Instead, deadlock avoidance mechanisms can be used to detect any deadlock situation in advance. Methods like "wait-for graph" are available but they are suitable for only those systems where transactions are lightweight having fewer instances of resource. In a bulky system, deadlock prevention techniques may work well.

	Wait/Die	Wound/Wait
Older process needs a resource held by younger process	Older process waits	Younger process dies
Younger process needs a resource held by older process	Younger process dies	Younger process waits

#### 4.6.1 Timestamp Ordering, Optimistic Techniques, Granularity of Data Items

##### Q11. Explain the Timestamp Ordering protocol.

*Ans :*

Timestamp is a unique identifier created by the DBMS to identify the relative starting time of a transaction. Typically, timestamp values are assigned in the order in which the transactions are submitted to the system. So, a timestamp can be thought of as the transaction start time. Therefore, time stamping is a method of concurrency control in which each transaction is assigned a transaction timestamp. Timestamps must have two properties namely

1. **Uniqueness** : The uniqueness property assures that no equal timestamp values can exist.
2. **monotonicity** : Monotonicity assures that timestamp values always increase.

Timestamp are divided into further fields:

1. Granule Timestamps
2. Timestamp Ordering
3. Conflict Resolution in Timestamps

##### 1. Granule Timestamps

Granule timestamp is a record of the timestamp of the last transaction to access it. Each granule accessed by an active transaction must have a granule timestamp.

A separate record of last Read and Write accesses may be kept. Granule timestamp may cause Additional Write operations for Read accesses if they are stored with the granules.

The problem can be avoided by maintaining granule timestamps as an in-memory table. The table may be of limited size, since conflicts may only occur between current transactions. An entry in a granule timestamp table consists of the granule identifier and the transaction timestamp. The record containing the largest (latest) granule timestamp removed from the table is also maintained. A search for a granule timestamp, using the granule identifier, will either be successful or will use the largest removed timestamp.

##### 2. Timestamp Ordering

Following are the three basic variants of timestamp-based methods of concurrency control:

- **Total timestamp ordering:** The total timestamp ordering algorithm depends on maintaining access to granules in timestamp order by aborting one of the transactions involved in any conflicting access. No distinction is made between Read and Write access, so only a single value is required for each granule timestamp.
- **Partial timestamp ordering:** In a partial timestamp ordering, only non-permutable actions are ordered to improve upon the total timestamp ordering. In this case, both Read and Write granule timestamps are stored. The algorithm allows the granule to be read by any transaction younger than the last transaction that updated the granule. A transaction is aborted if it tries to update a granule

that has previously been accessed by a younger transaction. The partial timestamp ordering algorithm aborts fewer transactions than the total timestamp ordering algorithm, at the cost of extra storage for granule timestamps

- **Multiversion Timestamp ordering:** The multiversion timestamp ordering algorithm stores several versions of an updated granule, allowing transactions to see a consistent set of versions for all granules it accesses. So, it reduces the conflicts that result in transaction restarts to those where there is a Write-Write conflict. Each update of a granule creates a new version, with an associated granule timestamp. A transaction that requires read access to the granule sees the youngest version that is older than the transaction. That is, the version having a timestamp equal to or immediately below the transaction's timestamp.

### Conflict Resolution in Timestamps

To deal with conflicts in timestamp algorithms, some transactions involved in conflicts are made to wait and to abort others. Following are the main strategies of conflict resolution in timestamps :

#### WAIT-DIE :

- The older transaction waits for the younger if the younger has accessed the granule first.
- The younger transaction is aborted (dies) and restarted if it tries to access a granule after an older concurrent transaction.

#### WOUND-WAIT :

- The older transaction pre-empt the younger by suspending (wounding) it if the younger transaction tries to access a granule after an older concurrent transaction.
- An older transaction will wait for a younger one to commit if the younger has accessed a granule that both want.

The handling of aborted transactions is an important aspect of conflict resolution algorithm. In the case that the aborted transaction is the one requesting access, the transaction must be restarted with a new (younger) timestamp. It is possible that

the transaction can be repeatedly aborted if there are conflicts with other transactions.

An aborted transaction that had prior access to granule where conflict occurred can be restarted with the same timestamp. This will take priority by eliminating the possibility of transaction being continuously locked out.

### Q12. Discuss in detail about multi version timestamp ordering.

*Ans :*

(Imp.)

#### Multi-version Timestamp Ordering

In this method for each object its temporary versions are maintained along with its committed versions. This maintenance avoid the need to reject late arriving read operations. Beside this, both read and write timestamps are maintained where a read timestamp contain maximum timestamp of the transaction which read the object.

Moreover, a read operation is directed to maximum write timestamp which has less timestamp than the transaction. When a transaction timestamp is greater than the version's read timestamp then the transaction timestamp uses the read timestamp of the version. In addition to this, a read operation is never aborted though it may sometime, made to wait to allow earlier transactions to complete their operation. Late arriving read operations are also permitted to read previous committed version in this method. This ensure recoverability of the execution. Furthermore, each transaction write its own committed versions of the object that it accesses. As a result of this, conflict occur among different transaction's write operations.

However, this method must follow a rule which states that a current transaction must not write objects, read by earlier transaction. In other words, if an object for any version has greater timestamp than current timestamp then this rule will be violated.

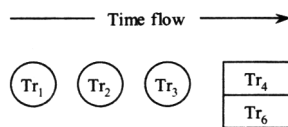
#### Multi-version Timestamp Ordering Write Rule

When a conflicting read operation is directed to the server, maximum write timestamp is checked and below stated rule is followed.

if  
 (an object O1's earlier maximum read timestamp is less than or equal to current timestamp)  
 then  
 perform a write on O1 's temporary version with current transaction's write timestamp  
 else  
 abort current transaction

### Example

Consider figure that shows sequence of operations that are requested to perform on an object.



Tr<sub>1</sub>, Tr<sub>2</sub> and Tr<sub>3</sub> are committed versions

Tr<sub>4</sub> and Tr<sub>6</sub> are temporary versions

Sequence = T<sub>4</sub> read, T<sub>1</sub> write, T<sub>6</sub> read, T<sub>5</sub> write

**Fig: A Late Write Operation Making a Read Operation Invalid**

In this figure, transaction4 (Tr<sub>4</sub>) needs to read from the object and place read timestamp of Tr<sub>4</sub> on transactions's version. After this, Tr<sub>4</sub> performs a write operation on the object and create a temporary version with Tr<sub>4</sub> write timestamp. Now, transaction6 (Tr<sub>6</sub>) performs a read operation on the same object which has Tr<sub>4</sub> write timestamp. Finally, transactions (Tr<sub>5</sub>) request to perform a write operation. This request is denied because the object has read timestamp of transaction6 which is greater than Tr<sub>5</sub>. In case, if this operation is allowed then the later write timestamp will be transaction (Tr<sub>5</sub>) and this will make transaction (Tr<sub>6</sub>)'s read operations invalid.

Furthermore, in this method of concurrency control when a transaction is committed, its all versions are stored whereas if a transaction is aborted its all versions are deleted. It also offers some advantages such as,

- (i) It allow more concurrency
- (ii) It is free from deadlock
- (iii) It never derives read operation.

However, it need more storage to save many versions.

## 4.7 DATABASE RECOVERY

### Q13. What is Database Recovery ?

*Ans :*

Data recovery is the art of restoring lost or damaged files. This damage can occur when your computer crashes, a virus infects, you accidentally reformat a disk that contains precious data, or you experience some other catastrophe of considerable dimension. And, at some point in your life, you're going to delete a file you really didn't mean to (believe me). The next time tragedy strikes, try running one of the many data recovery applications (powerful software written specifically for data recovery purposes) to see if it can correct the situation. Often these little jewels work magic and save your day-and your files.

There are companies that specialize in restoring lost data. These people generally work with more serious situations, like when you lose the last three years of your business's financial records. Find them in the yellow pages under "Computers: Service and Repair." Check the ads in the back pages of computer magazines if you can't find someone local. It's very common to have to ship your hard disk or your computer off to a company that can fix it.

#### 4.7.1 The Need for Recovery

### Q14. Why the need for recovery in databases and what is the role of transactions in database recovery ?

*Ans :*

A database is a very huge system with lots of data and transaction. The transaction in the database is executed at each seconds of time and is very critical to the database. If there is any failure or crash while executing the transaction, then it expected that no data is lost. It is necessary to revert the changes of transaction to previously committed point. There are various techniques to recover the data depending on the type of failure or crash.

#### 1. Transaction Failure

This is the condition in the transaction where a transaction cannot execute it further. This type of failure affects only few tables or



processes. The failure can be because of logical errors in the code or because of system errors like deadlock or unavailability of system resources to execute the transactions.

## 2. Crash Recovery

DBMS is a highly complex system with hundreds of transactions being executed every second. The durability and robustness of a DBMS depends on its complex architecture and its underlying hardware and system software. If it fails or crashes amid transactions, it is expected that the system would follow some sort of algorithm or techniques to recover lost data.

## 3. Failure Classification

To see where the problem has occurred, we generalize a failure into various categories, as follows

## 4. Transaction failure

A transaction has to abort when it fails to execute or when it reaches a point from where it can't go any further. This is called transaction failure where only a few transactions or processes are hurt. Reasons for a transaction failure could be -

- **Logical errors** - Where a transaction cannot complete because it has some code error or any internal error condition.
- **System errors** - Where the database system itself terminates an active transaction because the DBMS is not able to execute it, or it has to stop because of some system condition. For example, in case of deadlock or resource unavailability, the system aborts an active transaction.

## 5. System Crash

There are problems " external to the system " that may cause the system to stop abruptly and cause the system to crash. For example, interruptions in power supply may cause the failure of underlying hardware or software failure.

## 6. Disk Failure

In early days of technology evolution, it was a common problem where hard-disk drives or storage drives used to fail frequently. Disk failures include formation of bad sectors, unreachability to the disk, disk head crash or any other failure, which destroys all or a part of disk storage.

## Storage Structure

We have already described the storage system. In brief, the storage structure can be divided into two categories -

- **Volatile storage:** As the name suggests, a volatile storage cannot survive system crashes. Volatile storage devices are placed very close to the CPU; normally they are embedded onto the chipset itself. For example, main memory and cache memory are examples of volatile storage. They are fast but can store only a small amount of information.
- **Non-volatile storage:** These memories are made to survive system crashes. They are huge in data storage capacity, but slower in accessibility. Examples may include hard-disks, magnetic tapes, flash memory, and non-volatile (battery backed up) RAM.

## Q15. Explain the working mechanism of recovery.

*Ans :*

(Imp.)

The data recovery process varies, depending on the circumstances of the data loss, the data recovery software used to create the backup and the backup target media. For example, many desktop and laptop backup software platforms allow users to restore lost files themselves, while restoration of a corrupted database from a tape backup is a more complicated process that requires IT intervention. Data recovery services can also be used to retrieve files that were not backed up and accidentally deleted from a computer's file system, but still remain on the hard disk in fragments.

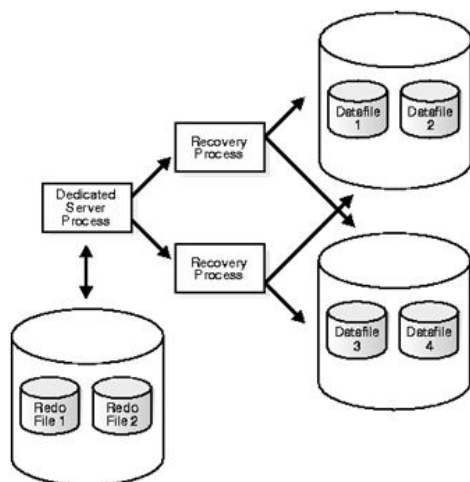
Data recovery is possible because a file and the information about that file are stored in different places. For example, the Windows operating

system uses a file allocation table to track which files are on the hard drive and where they are stored. The allocation table is like a book's table of contents, while the actual files on the hard drive are like the pages in the book.

When data needs to be recovered, it's usually only the file allocation table that's not working properly. The actual file to be recovered may still be on the hard drive in flawless condition. If the file still exists — and it is not damaged or encrypted — it can be recovered. If the file is damaged, missing or encrypted, there are other ways of recovering it. If the file is physically damaged, it can still be reconstructed. Many applications, such as Microsoft Office, put uniform headers at the beginning of files to designate that they belong to that application. Some utilities can be used to reconstruct the file headers manually, so at least some of the file can be recovered.

Most data recovery processes combine technologies, so organizations aren't solely recovering data by tape. Recovering core applications and data from tape takes time, and you may need to access your data immediately after a disaster. There are also risks involved with transporting tapes.

In addition, not all production data at a remote location may be needed to resume operations. Therefore, it's wise to identify what can be left behind and what data must be recovered.



#### 4.7.2 Transactions and Recovery, Recovery Facilities

##### Q16. Explain about Transactions Recovery.

*Ans :*

When a system fails, the database system is started again to perform the recovery actions, which are done in the following two phases,

##### (i) Redo Phase

In this phase, the scanning of the log is done in a forward manner from the most recent checkpoint so that, all the transaction updates done by the system are replayed. The replayed log records contain those records that were used in transaction rollbacks (before a failure occurred) and also those records that were not committed when a failure occurred. Generally, these records appear as  $\langle T_a, X_b, V_1, V_2 \rangle$  and  $\langle T_a, X_b, V_2 \rangle$ . Here,  $V_2$  is assigned to the data item  $X_b$ . The redo phase also finds the transactions that are executed later or those present in the transactions list available in the checkpoint records. These transactions contain no  $\langle T_a \text{ abort} \rangle$  or  $\langle T_a \text{ commit} \rangle$  record in the log. They should perform a rollback operation and their transaction identifiers must be placed in an undo-list by the system.

##### (ii) Undo Phase

In this phase, the rollback operation is performed on all the transactions present in the undo-list. Here, scanning is done in backward manner i.e., from the end of the log. If a log record of a transaction is found in the undo-list then undo operations are performed on those records that were identified at the time of failed transaction rollback. Hence, those log records that appear after the operation-begin record and before an operation-end record are skipped. The scanning of the log is continued until  $\langle T_a \text{ begin} \rangle$  record is found in the undo-list. Once the  $\langle T_a \text{ begin} \rangle$  log record has been found, the scanning is stopped and a  $\langle T_a \text{ abort} \rangle$  log record is written to the log.

In the redo phase, all the update operations that were executed by the transactions are replayed

from the last checkpoint. The update operations of those records that reached a stable log are also replayed. This include all those operations that were performed to rollback the failed transactions and also the operations that were executed to complete the partially completed transactions. The operations are repeated or replayed in similar manner in which they were performed. Hence, this process is known as a repeating history scheme. This scheme helps greatly in simplifying the other recovery schemes.

If a system failure occur during the processing of an undo operation then the physical log records that were written inside the log were identified. With the help of these records, partial undo operations were undone. While the system recovery is in progress, if the undo operation is completed, then the operation-end record is identified and the undo operation is performed again on the log records.

#### **Q17. How Media Failures effect Database Operation.**

*Ans :*

Media failures can affect one or all types of files necessary for the operation of an Oracle database, including datafiles, online redo log files, and control files.

Database operation after a media failure of online redo log files or control files depends on whether the online redo log or control file is multiplexed, as recommended. A multiplexed online redo log or control file means that a second copy of the file is maintained. If a media failure damages a single disk, and you have a multiplexed online redo log, the database can usually continue to operate without significant interruption. Damage to a non-multiplexed online redo log causes database operation to halt and may cause permanent loss of data. Damage to any control file, whether it is multiplexed or non-multiplexed, halts database operation once Oracle attempts to read or write the damaged control file, which happens frequently, for example at every checkpoint and log switch.

Media failures that affect datafiles can be divided into two categories: read errors and write errors. In a read error, Oracle discovers it cannot read a datafile and an operating system error is returned to the application, along with an Oracle error indicating that the file cannot be found, cannot

be opened, or cannot be read. Oracle continues to run, but the error is returned each time an unsuccessful read occurs. At the next checkpoint, a write error will occur when Oracle attempts to write the file header as part of the standard checkpoint process.

If Oracle discovers that it cannot write to a datafile and Oracle is in ARCHIVELOG mode, then Oracle returns an error in the DBWn trace file and takes the datafile offline automatically. Only the datafile that cannot be written to is taken offline; the tablespace containing that file remains online.

If the datafile that cannot be written to is in the SYSTEM tablespace, then the file is not taken offline. Instead, an error is returned and Oracle shuts down the instance. The reason for this exception is that all files in the SYSTEM tablespace must be online in order for Oracle to operate properly. For the same reason, the datafiles of a tablespace containing active rollback segments must remain online.

If Oracle discovers that it cannot write to a datafile, and Oracle is not archiving the filled online redo log files, then the DBWn background process fails and the current instance fails. If the problem is temporary (for example, the disk controller was powered off), then crash or instance recovery usually can be performed using the online redo log files, in which case the instance can be restarted. However, if a datafile is permanently damaged and archiving is not used, then the entire database must be restored using the most recent consistent backup.

#### **Recovery of Read-Only Tablespaces**

Recovery is not needed on read-only datafiles during crash or instance recovery. Recovery during startup verifies that each online read-only file does not need any media recovery. That is, the file was not restored from a backup taken before it was made read-only. If you restore a read-only tablespace from a backup taken before the tablespace was made read-only, then you cannot access the tablespace until you complete media recovery.

### Structures Used for Database Recovery

Several structures of an Oracle database safeguard data against possible failures. This section introduces each of these structures and its role in database recovery.

#### Database Backups

A database backup consists of backups of the physical files (all datafiles and a control file) that constitute an Oracle database. To begin media recovery after a media failure, Oracle uses file backups to restore damaged datafiles or control files. Replacing a current, possibly damaged, copy of a data file, table space, or database with a backup copy is called restoring that portion of the database.

Oracle offers several options in performing database backups, including :

- Recovery Manager
- Operating system utilities
- Export utility
- Enterprise Backup Utility

#### 4.7.3 Recovery Techniques

##### Q18. Explain different Recovery Techniques/ Approaches in DBMS ?

*Ans :* (Imp.)

**Database systems**, like any other computer system, are subject to failures but the data stored in it must be available as and when required. When a database fails it must possess the facilities for fast recovery. It must also have atomicity i.e. either transactions are completed successfully and committed (the effect is recorded permanently in the database) or the transaction should have no effect on the database.

There are both automatic and non-automatic ways for both, backing up of data and recovery from any failure situations. The techniques used to recover the lost data due to system crash, transaction errors, viruses, catastrophic failure, incorrect commands execution etc. are database recovery techniques. So to prevent data loss recovery techniques based on deferred update and immediate update or backing up data can be used.

Recovery techniques are heavily dependent upon the existence of a special file known as a system log. It contains information about the start and end of each transaction and any updates which occur in the transaction. The log keeps track of all transaction operations that affect the values of database items. This information is needed to recover from transaction failure.

- **The log is kept on disk start \_ transaction (T) :** This log entry records that transaction T starts the execution.
- **read\_item(T, X):** This log entry records that transaction T reads the value of database item X.
- **write\_item(T, X, old\_value, new\_value):** This log entry records that transaction T changes the value of the database item X from old\_value to new\_value. The old value is sometimes known as a before an image of X, and the new value is known as an afterimage of X.
- **commit(T) :** This log entry records that transaction T has completed all accesses to the database successfully and its effect can be committed (recorded permanently) to the database.
- **abort(T) :** This records that transaction T has been aborted.
- **checkpoint :** Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.

A transaction T reaches its commit point when all its operations that access the database have been executed successfully i.e. the transaction has reached the point at which it will not abort (terminate without completing). Once committed, the transaction is permanently recorded in the database. Commitment always involves writing a commit entry to the log and writing the log to disk.

At the time of a system crash, item is searched back in the log for all transactions T that have written a start\_transaction(T) entry into the log but have not written a commit(T) entry yet; these transactions may have to be rolled back to undo their effect on the database during the recovery process

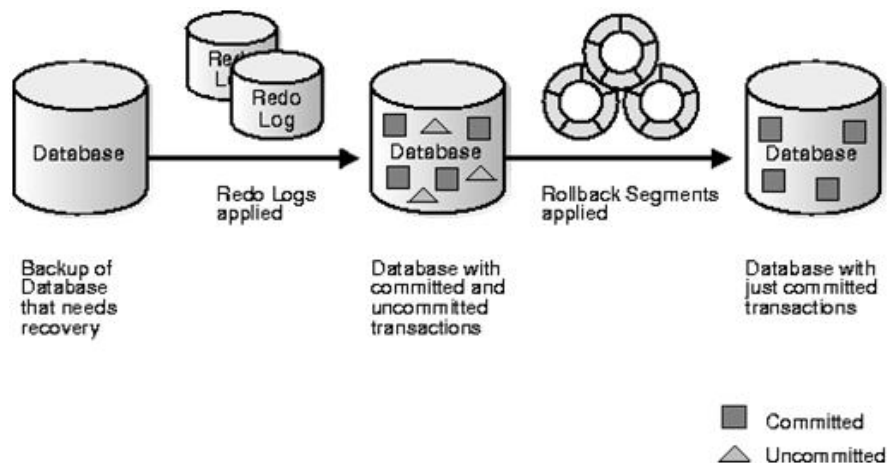
- **Undoing:** If a transaction crashes, then the recovery manager may undo transactions i.e. reverse the operations of a transaction. This involves examining a transaction for the log entry write\_item(T, x, old\_value, new\_value) and setting the value of item x in the database to old-value. There are two major techniques for recovery from non-catastrophic transaction failures: deferred updates and immediate updates.
- **Deferred update:** This technique does not physically update the database on disk until a transaction has reached its commit point. Before reaching commit, all transaction updates are recorded in the local transaction workspace. If a transaction fails before reaching its commit point, it will not have changed the database in any way so UNDO is not needed. It may be necessary to REDO the effect of the operations that are recorded in the local transaction workspace, because their effect may not yet have been written in the database. Hence, a deferred update is also known as the No-undo/redo algorithm
- **Immediate update:** In the immediate update, the database may be updated by some operations of a transaction before the transaction reaches its commit point. However, these operations are recorded in a log on disk before they are applied to the database, making recovery still possible. If a transaction reaches its commit point, the effect of its operation must be undone i.e. the transaction must be rolled back hence we require both undo and redo. This technique is known as undo/redo algorithm.
- **Caching/Buffering:** In this one or more disk pages that include data items to be updated

are cached into main memory buffers and then updated in memory before being written back to disk. A collection of in-memory buffers called the DBMS cache is kept under control of DBMS for holding these buffers. A directory is used to keep track of which database items are in the buffer. A dirty bit is associated with each buffer, which is 0 if the buffer is not modified else 1 if modified.

- **Shadow paging:** It provides atomicity and durability. A directory with n entries is constructed, where the ith entry points to the ith database page on the link. When a transaction began executing the current directory is copied into a shadow directory. When a page is to be modified, a shadow page is allocated in which changes are made and when it is ready to become durable, all pages that refer to original are updated to refer new replacement page.

Some of the backup techniques are as follows :

- **Full database backup:** In this full database including data and database, Meta information needed to restore the whole database, including full-text catalogs are backed up in a predefined time series.
- **Differential backup:** It stores only the data changes that have occurred since last full database backup. When same data has changed many times since last full database backup, a differential backup stores the most recent version of changed data. For this first, we need to restore a full database backup.
- **Transaction log backup:** In this, all events that have occurred in the database, like a record of every single statement executed is backed up. It is the backup of transaction log entries and contains all transaction that had happened to the database. Through this, the database can be recovered to a specific point in time. It is even possible to perform a backup from a transaction log if the data files are destroyed and not even a single committed transaction is lost.



#### 4.7.4 Nested Transaction Model

**Q19. Explain about Nested Transaction Model.**

*Ans :*

##### **Nested Transaction Model**

A nested transaction refers to a transaction which contains sub-transactions. There by forming a tree structure. In this tree structure, the transactions residing at the higher levels act as parent transactions of their lower level transactions. According to Moss proposal, only leaf nodes i.e., the nodes at the lowest level can perform operations.

##### **Example**

```

begin_t1
  begin_t2
    orderabook;
    commit t2;
  begin_t3
    begin_t4
      searchforabook;
      commit t4;
    commit t3;
  begin_t5 cancelanorder;
  commit t5;
commit t1;

```

In the above example, the leaf transactions i.e., t2, t4 and t5 performs various operations. The transaction about at lower level does not necessarily affect the transactions at higher levels. In case of a sub-transaction failure, parent transaction can perform any of the following operations.

- Restart the transaction.
- Simply ignore the failure.

- Start a different transaction in place of the failed transaction.
- Terminate the transaction.

The modification performed on the database by various sub-transactions are reflected only within their parent region. Therefore, modifications performed by t4 can only be visible to t3. With such an approach, the higher-level transactions can achieve ACID properties.

To provide concurrency control in this model, strict-two phase locking mechanism can be used. Using this mechanism, sub-transactions are executed as individual transactions while carrying a lock. This lock is inherited by its parent whenever the transaction is completed successfully.

#### Advantages of Nested Transaction Model

- It provides modularity by allowing nested transactions.
- It provides finer level of granularity by allowing concurrency control and recovery to be performed at sub-transaction level.
- It offers infra-transaction recovery with which transactions can be rolled back or terminated without effecting others.
- It provides concurrent execution of subtransactions.

### 4.8 DATABASE SECURITY

#### Q20. Explain the Concepts behind the Data Security?

*Ans :*

Threats and risks to databases have increased and therefore, the need for securing databases has also increased. Let's learn about the basic facets of database security, including assurance, integrity, availability, and confidentiality.

The majority of the companies store sensitive data in databases. However, database security is sometimes not given as much thought and effort as other areas of computer security. Hackers have been able to target large databases in recent years to obtain sensitive information like credit card numbers

and other personal information. It is important to protect databases against these risks, and this is where database security comes into place.

Database security can be defined as a system or process by which the "Confidentiality, Integrity, and Availability," or CIA, of the database can be protected. Unauthorized entry or access to a database server signifies a loss of confidentiality; unauthorized alteration to the available data signifies loss of integrity; and lack of access to database services signifies loss of availability. Loss of one or more of these basic facets will have a significant impact on the security of the database.

For an illustration of this concept, imagine that the website of a company contains information like who they are, what they do, and what prospective customers have to do to contact them for their queries. In this case, the availability of the database services is more important when compared with other factors like the confidentiality or integrity of the database security.

For a company that sells products or goods online, however, confidentiality and integrity are more important as customers use their credit cards to buy goods online only when the site is available.

Another factor needs to be addressed when examining database security and that is "Assurance."

What is database assurance? Take for example, a web application that acts as a frontend to a database server. If the web application that is selling online goods is vulnerable to cross-site-scripting, the chances of people not trusting the website becomes greater. When customers lose trust or assurance in the company, this may consequently lead to loss in the business.

Databases are susceptible to other vulnerabilities like poor password management, SQL injection, leakage of data, and improper error handling apart from cross site scripting. Hackers try to attack databases that are configured poorly. Hackers take advantage of these database weaknesses to exploit the database vulnerabilities.

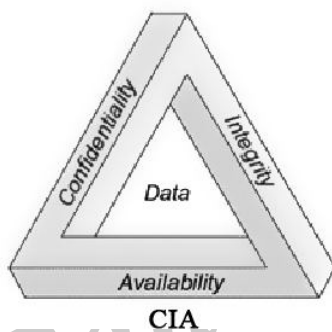
The risks involved with the database are not the same for every database present in the organization; therefore, security controls or measures

to these databases differ. As there are various databases like Oracle, SQL and Access, different types of database security solutions are also available in the market. One needs to assess the risk for the database involved and mitigate the risk by designing and implementing appropriate database security solutions. If security is the key driver for database configuration, the data will remain safe and secure

### Concepts of Database Security

When it comes to securing a database, lots of things have to be taken care of. This article provides details about core database security concepts including confidentiality, integrity, and availability.

In the first article in this series, we saw what database security is and introduced the core concepts of database security. In this part, we'll continue our look, in somewhat greater detail, at the core concepts that play a vital role in database security.



### Confidentiality

In database security concepts, Confidentiality comes first. Confidentiality can be enforced by encrypting the data stored in the database. Encryption is a technique or process by which data is encoded in such a way only authorized users be able to read the data. In other words, encryption means rendering sensitive data unreadable to unauthorized users. Encryption can be done at two different levels: data-in-transit and data-at-rest.

- **Data-in-transit** : This refers to data that is moving within the network. Sensitive data, for example, that is sent through network layers or through the Internet. A hacker can gain access to this sensitive data by eavesdropping. When this happens, the confidentiality of the data is compromised.

Encrypting data-in-transit avoids such compromises.

- **Data-at-rest** : It is possible for a hacker to hack the data that is stored in the database. Encrypting data-at-rest prevents such data leakages.

Different encryption algorithms are available, which includes Data Encryption Standards (DES), Triple DES or 3DES, and Advanced Encryption Standards (AES).

### Integrity

Integrity can be enforced by setting User Access Controls (UAC) that define which users have to be given what permissions in the database. For example, data related to employee information is stored in a database. An employee may have permission for viewing the records and altering only part of information like his contact details, whereas a person in the human resources department will have more privileges.

What are the steps that have to be taken to ensure integrity of the database ?

- Once the database is installed, the password has to be changed. Similarly, periodic checks have to be conducted to ensure the password is not compromised.
- User accounts that are not in use have to be locked. If one is sure that these user accounts will never be used again, then the best step is to remove such user accounts.

### 4.8.1 Threats to Database Security

#### Q21. Explain different Database Security Threats and How to Mitigate them?

*Ans :*

(Imp.)

With the increase in usage of databases, the frequency of attacks against those databases has also increased. Here we look at some of the threats that database administrators actually can do something about.

Database attacks are an increasing trend these days. What is the reason behind database attacks?



that are related to the databases could lead to unauthorized access. This may lead to a Denial of Service (DoS) attack. This could be prevented by updating the operating system related security patches as and when they become available.

### 3. Database Rootkits

## 1. Privilege abuse

#### 4. Weak authentication

Weak authentication models allow attackers to employ strategies such as social engineering and brute force to obtain database login credentials and assume the identity of legitimate database users.

## 5. Weak audit trails

203

#### 4.8.2 Computer Based Controls

##### Q22. Explain various Computer Based Controls for a multi user environment

*Ans :* (Imp.)

Various computer-based security controls for a multi-user environment are,

1. Access control
2. View control
3. Backup and recovery
4. Authentication/authorization
5. Integrity
6. Encryption
7. RAID.

#### 1. Access Control

Access control is a way of prohibiting unauthorized users from data access. The two most common access control mechanisms are,

- (i) Discretionary Access Control (DAC)
- (ii) Mandatory Access Control (MAC).

#### (i) Discretionary Access Control (DAC)

DAC specifies that the users must not just protect their own files but also allow or deny access rights to other users or group of users. In addition to this, it also specifies access mode.

#### (ii) Mandatory Access Control (MAC)

MAC specifies that, users or group of users can perform both read and write operations on files. The functioning of MAC is done according to the security labels which are allocated to both users and files. These security labels are nothing but hierarchical classification level and set of non-hierarchical categories. They together states the privileges of user and shows how critical the file information is? In particular, users in MAC can read the files from same classification or also from lower classification. Similarly, it can write on files from same classification or from higher classification.

#### 2. View Control

View is another way of ensuring data security. It puts restriction on the users such that they are not allowed to view the entire data rather allows them to view only the part of data for which they have been authorized.

Generally, views are represented using relational operators such as selections projections and joins. Using views, restrictions are provided on all existing relations or executing joins on certain relations thereby ensuring data security.

Views are created for different classes of users due to which it is possible to automatically attain access controls. Moreover relational-level-security and view-level-security can be integrated for achieving high level of access control such that the users are allowed to access only required data.

#### 3. Backup and Recovery

Backup and recovery control is important in database system for operations to be performed efficiently. This control is useful to restore the database during system crash.

##### Backups

There are three different levels of database backup. They are,

#### (i) Full Backup

In this level, all the contents of the database are copied to a permanent storage device. For example disks, tapes etc.

#### (ii) Differential Backup

In this level, only those contents of the database that have been modified, as compared to previous backup copy, are copied to a permanent secondary storage.

#### (iii) Transaction Log Backup

In this level, only those transaction log operations that have been modified, as compared to the previous backup copy, are copied to a permanent secondary storage.

##### Recovery

In database recovery process, database backups play a major role. The database recovery process chooses to utilize any one of the levels of

backup, depending on the type and extent of recovery that is to be performed. If a whole database is to be recovered, then the recovery process involves loading of the recent backup copy of the database, when it was in a consistent state. Then, the transaction log information is used to roll forward the backup copy in order to restore all the remaining transactions carried out after database has moved from the consistent state. If these transactions have been committed and are still usable, then the recovery process can use the transaction log information and perform just an 'undo' operation on all those transactions that were not committed.

#### 4. Authentication/Authorization

Authentication is a mean of verifying user's identity. This is done by using any of the following mechanisms,

- (i) Username and password
- (ii) Plastic ID card
- (iii) Physical representation like finger recognition (or) voice recognition techniques.

Among these methods, password is one of the cheaper source to provide security. But, when intruder knows the password length and the characters available, it is possible to hack the password. So, DBA must design excellent password schemes to avoid hacking.

One of the typical authorization scheme is to create a password which is easy for the users to remember and at the same time difficult for the intruders to guess. This scheme is capable of protecting the data from being accessed by unauthorized users.

Apart from this, the other schemes for ensuring authentication are as follows,

- (a) Usage of password parameters
- (b) Question-and-answer scheme
- (c) Usage of pre-arranged algorithm
- (d) Usage of access control mechanism.

#### 5. Integrity

Integrity constraint is defined as the rule that set various restrictions on certain records of the database.

#### 6. Encryption

It is the process of encoding information in such a way that only authorized users (who knows the method of encryption) have ability to read the information. Intruders cannot read the encrypted data until the encryption method is known.

There are two encryption methods used for encrypting the data. They are,

- (a) Simple substitution method
- (b) Poly-alphabetic substitution method

##### (a) Simple Substitution Method

It is a method in which each character is substituted by another character.

##### (b) Poly-alphabetic substitution method

Polyalphabetic ciphers are the improved form of mono-alphabetic ciphers, that uses different mono-alphabetic ciphers for different letters in the plaintext.

The common features of this technique are,

- 1. Using a related set of mono-alphabetic substitution rules.
- 2. For a given transformation, the specific rule to be used is determined by the key.

The simplest algorithm that contains the above features is a vigenere cipher. Here, the set of monoalphabetic rules that are related to each other are the 26 Caesar ciphers that have the shifts from 0 through 25. Each of the 26 ciphers are placed horizontally with their key letter to its left. The normal plaintext letters are placed on the top. The process proceeds as follows,

If 'r' is a key letter and 'h' is a plaintext letter, then the intersection of row r and column h is 'y' and is the corresponding ciphertext letter.

Encryption of a message requires a key, that is of the length of the message. The strength of encryption depends on the type of the key used.

#### 7. RAID

RAID stands for Redundant Array of Inexpensive Disks. The idea here is to use multiple disks and keep redundant data or copies of data so as to improve performance and reliability. The

simplest RAID is to copy a whole disk to another disk. Thus, if the original disk fails, its duplicate can be used to restore the data. Hence, reliability is increased.

Here both the original and duplicate disks can be used concurrently to access data. Consider an example, that we want to read 16-blocks of data from a disk and it requires 16 milliseconds. If we read first eight blocks from one disk and remaining eight from another simultaneously, then the time required will be only 8 milliseconds i.e., half of the traditional approach. Hence, performance is improved.

#### 4.9 AUTHORIZATION

##### Q23. Explain the concept of Authorization.

*Ans :*

Authorization is a security mechanism used to determine user/client privileges or access levels related to system resources, including computer programs, files, services, data and application features. Authorization is normally preceded by authentication for user identity verification. System administrators (SA) are typically assigned permission levels covering all system and user resources.

During authorization, a system verifies an authenticated user's access rules and either grants or refuses resource access.

Modern and multiuser operating systems depend on effectively designed authorization processes to facilitate application deployment and management. Key factors include user type, number, and credentials requiring verification and related actions and roles. For example, role-based authorization may be designated by user groups requiring specific user resource tracking privileges. Additionally, authorization may be based on an enterprise authentication mechanism, like Active Directory (AD), for seamless security policy integration.

For example, ASP.NET works with Internet Information Server (IIS) and Microsoft Windows to provide authentication and authorization services for Web-based .NET applications. Windows uses New Technology File System (NTFS) to maintain Access Control Lists (ACL) for all resources. The

ACL serves as the ultimate authority on resource access.

The .NET Framework provides an alternate role-based security approach for authorization support. Role-based security is a flexible method that suits server applications and is similar to code access security checks, where authorized application users are determined according to roles.

- Authorization is finding out if the person, once identified, is permitted to have the resource.
- Authorization explains that what you can do and is handled through the DBMS unless external security procedures are available.
- Database management system allows DBA to give different access rights to the users as per their requirements.
- Basic Authorization we can use any one form or combination of the following basic forms of authorizations.

1. **Resource authorization:** Authorization to access any system resource. e.g. sharing of database, printer etc.
2. **Alteration Authorization:** Authorization to add attributes or delete attributes from relations
3. **Drop Authorization:** Authorization to drop a relation.

##### Granting of Privileges

- i. A system privilege is the right to perform a particular action, or to perform an action on any schema objects of a particular type.
- ii. An authorized user may pass on this authorization to other users. This process is called as granting of privileges.

##### Syntax :

GRANT <privilege list> ON <relation name or view name> TO <user/role list>

##### Example:

The following grant statement grants user U1, U2 and U3 the select privilege on Emp\_Salary relation:

GRANT select ON Emp\_Salary TO U1, U2 and U3.

### Revoking of Privileges

We can reject the privileges given to particular user with help of revoke statement.

- To revoke an authorization, we use the revoke statement.

#### Syntax:

```
REVOKE <privilege list> ON <relation name or view name> FROM <user/role list> [restrict/cascade]
```

#### Example :

The revocation of privileges from user or role may cause other user or roles also have to loose that privileges. This behavior is called cascading of the revoke.

```
Revoke select ON Emp_Salary FROM U1,U2,U3.
```

### Some Other Types of Privileges

1. **Reference privileges :** SQL permits a user to declare foreign keys while creating relations.

**Example:** Allow user U1 to create relation that references key 'Eid' of Emp\_Salary relation.

```
GRANT REFERENCES(Eid) ON Emp_Salary TO U1
```

### Execute Privileges

This privilege authorizes a user to execute a function or procedure. Thus, only user who has execute privilege on a function Create\_Acc() can call function.

```
GRANT EXECUTE ON Create_Acc TO U1.
```

Using authentication, authorization, and encryption.

Authentication, authorization, and encryption are used in everyday life. One example in which authorization, authentication, and encryption are all used is booking and taking an airplane flight.

- Encryption is used when a person buys their ticket online at one of the many sites that advertises cheap ticket. Upon finding the perfect flight at an ideal price, a person goes to buy the ticket. Encryption is used to protect a person's credit card and personal information when it is sent over the Internet to the airline. The company encrypts the customer's data so that it will be safer from interception in transit.
- Authentication is used when a traveler shows his or her ticket and driver's license at the airport so he or she can check his or her bags and receive a boarding pass. Airports need to authenticate that the person is who he or she says she is and has purchased a ticket, before giving him or her a boarding pass.
- Authorization is used when a person shows his or her boarding pass to the flight attendant so he or she can board the specific plane he or she is supposed to be flying on. A flight attendant must authorize a person so that person can then see the inside of the plane and use the resources the plane has to fly from one place to the next.
- Here are a few examples of where encryption, authentication, and authorization are used by computers:
- Encryption should be used whenever people are giving out personal information to register for something or buy a product. Doing so ensures the person's privacy during the communication. Encryption is also often used when the data returned by the server to the client should be protected, such as a financial statement or test results.

- Authentication should be used whenever you want to know exactly who is using or viewing your site. Weblog in is Boston University's primary method of authentication. Other commercial websites such as Amazon.com require people to login before buying products so they know exactly who their purchasers are.
- Authorization should be used whenever you want to control viewer access of certain pages. For example, Boston University students are not authorized to view certain web pages dedicated to professors and administration. The authorization requirements for a site are typically defined in a website's .htaccess file.
- Authentication and Authorization are often used together. For example, students at Boston University are required to authenticate before accessing the Student Link. The authentication they provide determines what data they are authorized to see. The authorization step prevents students from seeing data of other students.

#### Q24. Explain managing and controlling the Data in DBMS using Roles ?

*Ans :*

Managing and controlling privileges is made easier by using **roles**, which are named groups of related privileges that you grant as a group to users or other roles. Within a database, each role name must be unique, different from all user names and all other role names. Unlike schema objects, roles are not contained in any schema. Therefore, a user who creates a role can be dropped with no effect on the role.

Roles are designed to ease the administration of an end-user system and schema object privileges and are often maintained in Oracle Internet Directory. However, roles are not meant to be used by application developers, because the privileges to access schema objects within stored programmatic constructs need to be granted directly.

The effective management of roles is discussed in the following subsections :

Authentication Considerations in These Topical Areas	Links to Relevant Subsection
Why Roles Are Advantageous	Properties of Roles
How Roles are Typically Used	Common Uses of Roles
How Users Get Roles (or Role Restrictions)	Granting and Revoking Roles
How Roles Affect The Scope of a User's Privileges	Security Domains of Roles and Users
How Roles Work in PL/SQL Blocks	PL/SQL Blocks and Roles
How Roles Aid or Restrict DDL Usage	DDL Statements and Roles
What Roles are Predefined in Oracle	Predefined Roles
How Can Operating Systems Aid Roles	Operating System and Roles
How Roles Work in a Remote Session	Roles in a Distributed Environment
How Secure Application Roles Are Created and Used	Secure Application Roles

### Properties of Roles

Property	Description
Reduced privilege Administration	Rather than granting the same set of privileges explicitly to several users, you can grant the privileges for a group of related users to a role, and then only the role needs to be granted to each member of the group.
Dynamic privilege Management	If the privileges of a group must change, then only the privileges of the role need to be modified. The security domains of all users granted the group's role automatically reflect the changes made to the role.
Selective availability of privileges	You can selectively enable or disable the roles granted to a user. This allows specific control of a user's privileges in any given situation.
Application awareness	The data dictionary records which roles exist, so you can design applications to query the dictionary and automatically enable (or disable) selective roles when a user attempts to execute the application by way of a given user name.
Application-specific security	You can protect role use with a password. Applications can be created specifically to enable a role when supplied the correct password. Users cannot enable the role if they do not know the password.

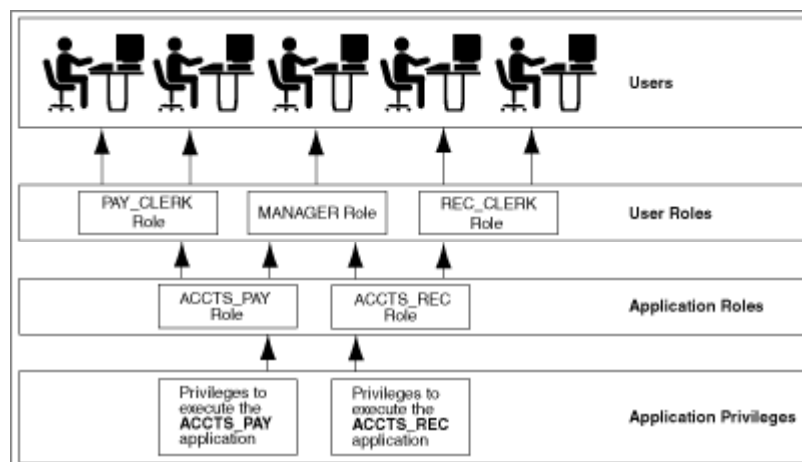
Database administrators often create roles for a database application. The DBA grants a secure application role all privileges necessary to run the application. The DBA then grants the secure application role to other roles or users. An application can have several different roles, each granted a different set of privileges that allow for more or less data access while using the application.

The DBA can create a role with a password to prevent unauthorized use of the privileges granted to the role. Typically, an application is designed so that when it starts, it enables the proper role. As a result, an application user does not need to know the password for an application role.

### Common Uses of Roles

In general, you create a role to serve one of two purposes :

- To manage the privileges for a database application (Application Roles)
- To manage the privileges for a user group (User Roles)



## Application Roles

You grant an application role all privileges necessary to run a given database application. Then, you grant the secure application role to other roles or to specific users. An application can have several different roles, with each role assigned a different set of privileges that allow for more or less data access while using the application.

## User Roles

You create a user role for a group of database users with common privilege requirements. You manage user privileges by granting secure application roles and privileges to the user role and then granting the user role to appropriate users.

## Granting and Revoking Roles

System or schema object privileges can be granted to a role, and any role can be granted to any database user or to another role (but not to itself). However, a role cannot be granted circularly, that is, a role X cannot be granted to role Y if role Y has previously been granted to role X.

To provide selective availability of privileges, Oracle Database allows applications and users to enable and disable roles. Each role granted to a user is, at any given time, either enabled or disabled. The security domain of a user includes the privileges of all roles currently enabled for the user and excludes the privileges of any roles currently disabled for the user.

A role granted to a role is called an indirectly granted role. It can be explicitly enabled or disabled for a user. However, whenever you enable a role that contains other roles, you implicitly enable all indirectly granted roles of the directly granted role.

You grant roles to (or revoke roles from) users or other roles by using either of the following methods :

- Oracle Enterprise Manager 10g Database Control
- The SQL statements, GRANT and REVOKE

Privileges are granted to and revoked from roles using the same options. Roles can also be granted to and revoked from users using the operating system that runs Oracle, or through network services.

## 4.10 ACCESS CONTROL MECHANISMS

### Q25. What are the types of Access Control Mechanisms ?

*Ans :*

(Imp.)

Many of us have come across the terms like MAC, DAC, RBAC, ACLs while reading various e-security related articles. However not all of us (except the CISSPs) know the meanings of these terms and the differences between these Access Control mechanisms.

Before proceeding to Access Control mechanisms, let's see what Access Control is.

Access Control is a set of controls to restrict access to certain resources. If we think about it, access controls are everywhere around us. A door to your room, the guards allowing you to enter the office building on seeing your access card, swiping your card and scanning your fingers on the biometric system, a queue for food at the canteen or entering your credentials to access FB, all are examples of various types of access control. Here we focus only on the logical Access Control mechanisms.

### Access Control Mechanisms

- **Discretionary Access Control (DAC):** As the name suggests, this access control model is based on a user's discretion. i.e, the owner of the resource can give access rights on that resource to other users based on his discretion. Access Control Lists (ACLs) are a typical example of DAC. Specifying the "rwx" permissions on a unix file owned by you is another example of DAC. Most of the operating systems including windows, flavours of unix are based on DAC Model.
- **Mandatory Access Control (MAC):** In this Model, users/owners do not enjoy the privilege of deciding who can access their files. Here the operating system is the decision maker overriding the user's wishes. In this model every Subject (users) and Object (resources) are classified and assigned with a security label. The security labels of the subject and the object along with the security policy determine if the subject can access the object.



The rules for how subjects access objects are made by the security officer, configured by the administrator, enforced by the operating system, and supported by security technologies. This is a stricter and rather static Access Control model as compared to DAC and is mostly suited for military organizations where data classification and confidentiality is of prime importance. Special types of the Unix operating systems are based on MAC model.

- **Role Based Access Control (RBAC):** RBAC is the buzzword across enterprises today. In this model the access to a resource is governed based on the role that the subject holds within an organization. RBAC is also known as non-discretionary Access Control because the user inherits privileges that are tied to his role. The user does not have a control over the role that he will be assigned. Each of the above Access Models has its own advantages and disadvantages. The selection of the appropriate Access Model by an organization should be done by considering various factors such as type of business, no of users, organization's security policy etc.

#### Case for RBAC

For implementing any access control, the two driving factors are :

- Least privilege principle i.e, the user should only have the minimum privileges to perform the tasks that he is supposed to do.
- Segregation of duties i.e, having more than one user to perform a critical task so as to reduce the risk of internal frauds.

As the no of users and the resources grow in an organization, it becomes extremely difficult to manage user's access rights through ACLs. It not only increases the cost of administration but also results in granting of excess privileges to users thus violating the least privilege principle and hence exposing the organization to risks. Moreover, the complexity involved with this approach makes it too hard for any organization to comply with the regulatory compliances.

So in organizations where the no of users and the employee turnover is large, RBAC is the optimum solution for Access Control. By having privileges tied to roles, and users being assigned to these roles, makes it much simpler for an organization to manage the access to its resources. RBAC also fastens the employee on-boarding & de-boarding process by tying the provisioning/de-provisioning to the roles.

Moreover, RBAC also makes the implementation of Least privilege principle and SOD easier, hence helping the organizations in complying to the strict regulatory standards.

RBAC but the roles itself are vaguely defined? Or if proper consideration is not given to the privileges being assigned to Roles. The whole purpose of adopting RBAC stands defeated in this case. This is where Role Engineering comes into play. I thought before heading to role engineering it's important that the reader is aware of a few basic concepts related to access control.

### 4.11 VIEWS

#### Q26. What is a view in DBMS ?

*Ans :*

A **view** is a virtual or logical table that allows to view or manipulate parts of the tables. To reduce REDUNDANT DATA to the minimum possible, Oracle allows the creation of an object called a VIEW.

A View is mapped, to a SELECT sentence. The table on which the view is based is described in the FROM clause of the SELECT statement.

Some Views are used only for looking at table data. Other Views can be used to Insert, Update and Delete table data as well as View data. If a View is used to only look at table data and nothing else the View is called a Read-Only View. A View that is used to look at table data as well as Insert, Update and Delete table data is called an Updateable View.

The reasons why views are created are :

- When Data security is required .
- When Data redundancy is to be kept to the minimum while maintaining data security.

### Types of Views

- (i) **Read-only View** : Allows only SELECT operations.
- (ii) **Updateable View** : Allows SELECT as well as INSERT, UPDATE and DELETE operations.

### Creating a View

The ORDER BY clause cannot be used while creating a view. The columns of the table are related to the view using a one-to-one relationship.

#### Syntax :

```
CREATE <ORREPLACE> VIEW <ViewName> AS SELECT <ColumnName1> , <Column Name2>  
FROM <TableName> WHERE <Column
```

Name > = < Expression List > < WITHREADONLY>;

This statements creates a view based on query specified in SELECT statement.

OR REPLACE option recreates the view if it is already existing maintaining the privileges granted to view viewname.

### Updateable Views

Views can also be used for data manipulation. Views on which data manipulation can be done are called Updateable Views.

When an updateable view name is given in an Insert Update, or Delete SQL statement, modifications to data in the view will be immediately passed to the underlying table.

For a view to be updateable, it should meet the following criteria :

- Views defined from Single table
- If the user wants to INSERT records with the help of a view, then the PRIMARY KEY column(s) and all the NOT NULL columns must be included in the view.
- The user can UPDATE, DELETE records with the help of a view even if the PRIMARY KEY column and NOT NULL column(s) are excluded from the view definition.

---

### Q27. Explain different types of Views ?

Ans :

#### 1. Horizontal view

A horizontal view cuts the source tables horizontally to create the view. It creates a horizontal view of the source tables. It is used when different names have to be displayed in a view.

##### For Example :

```
CREATE VIEW NORTHCOM AS SELECT * FROM COMPANIES WHERE REGION = 'Northern'
```

The above query creates a view having companies in northern region.

#### 2. Vertical view

A horizontal view cuts the source tables vertically to create the view.

##### For Example:

```
CREATE VIEW EMPLINFO AS SELECT NAME, EMPL_NUM, EMPL_OFFICE FROM EMPLOYEES
```

The above query creates a view EMPLINFO having name, employee number and office.

### 3. Row and column views

A row and column view uses both rows and columns to create the view.

**For Example:**

```
CREATE VIEW SPEMPL AS SELECT COMPANY, ORDER_NUM FROM ORDERS WHERE EMPL_NUM = 665
```

The above query returns all order numbers and company information for employee number 665.

### 4. Grouped by view

In a grouped by view, a GROUP BY clause is used. A grouped by view always includes a column list.

**For Example:**

```
CREATE VIEW SALPERKS (WHO, PERKS_LIST, AVG_SALARY) AS SELECT NAME, PERKS, AVG(SALARY) FROM EMPLOYEES GROUP BY NAME where, (WHO, PERKS_LIST, AVG _SALARY) is the column list in view;
```

### 5. Joined view

A joined view is used to create a multi table query for a view. The source of this type of view can be two or more tables.

**For Example:**

```
CREATE VIEW ORDERIN (ORDER_NUM, COMPANY, EMPL_NAME, PRICE) AS SELECT ORDER_NUM, COMPANY, NAME, PRICE FROM ORDERS, CLIENTS, EMPLOYEES WHERE CLIENT = CLIENT_NUM AND EMPL = EMPL_NUM;
```

---

#### 4.11.1 Backup and Recovery, Integrity

##### Q28. Explain briefly about Backup and Recovery and Integrity

*Ans :*

(Imp.)

##### Backup and Recovery

Backup and recovery control is important in database system for operations to be performed efficiently. This control is useful to restore the database during system crash.

##### Backups

There are three different levels of database backup. They are,

##### (i) Full Backup

In this level, all the contents of the database are copied to a permanent storage device. For example disks, tapes etc.

##### (ii) Differential Backup

In this level, only those contents of the database that have been modified, as compared to previous backup copy, are copied to a permanent secondary storage.

##### (iii) Transaction Log Backup

In this level, only those transaction log operations that have been modified, as compared to the previous backup copy, are copied to a permanent secondary storage.

## Recovery

In database recovery process, database backups play a major role. The database recovery process chooses to utilize any one of the levels of backup, depending on the type and extent of recovery that is to be performed. If a whole database is to be recovered, then the recovery process involves loading of the recent backup copy of the database, when it was in a consistent state. Then, the transaction log information is used to roll forward the backup copy in order to restore all the remaining transactions carried out after database has moved from the consistent state. If these transactions have been committed and are still usable, then the recovery process can use the transaction log information and perform just an 'undo' operation on all those transactions that were not committed.

## 5. Integrity

Integrity constraint is defined as the rule that set various restrictions on certain records of the database.

### 4.12 DATABASE ENCRYPTION AND DECRYPTION

**Q29. Explain the concept of Database Encryption and Decryption.**

*Ans :*

(Imp.)

Database encryption is the process of converting data, within a database, in plain text format into a meaningless cipher text by means of a suitable algorithm.

Database decryption is converting the meaningless cipher text into the original information using keys generated by the encryption algorithms.

Database encryption can be provided at the file or column level.

Encryption of a database is costly and requires more storage space than the original data. The steps in encrypting a database are :

1. Determine the criticality of the need for encryption
2. Determine what data needs to be encrypted
3. Determine which algorithms best suit the encryption standard
4. Determine how the keys will be managed.

Numerous algorithms are used for encryption. These algorithms generate keys related to the encrypted data. These keys set a link between the encryption and decryption procedures. The encrypted data can be decrypted only by using these keys.

Different databases, such as SQL, Oracle, Access and DB2, have unique encryption and decryption methods.

## Techniques used for Encryption

There are following techniques used for encryption process :

### 1. Substitution Ciphers

In a substitution cipher each letter or group of letters is replaced by another letter or group of letters to mask them. For example: a is replaced with D, b with E, c with F and z with C. In this way attack becomes DWWDFN. The substitution ciphers are not much secure because intruder can easily guess the substitution characters.

### 2. Transposition Ciphers

Substitution ciphers preserve the order of the plaintext symbols but mask them;-The transposition cipher in contrast reorders the letters but do not mask them. For this process a key is used. For

example: iliveinqadianmay be coded as divienaniqnli. The transposition ciphers are more secure as compared to substitution ciphers.

### Algorithms for Encryption Process

There are commonly used algorithms for encryption process. These are:

#### Data Encryption Standard (DES)

It uses both a substitution of characters and a rearrangement of their order on the basis of an encryption key. The main weakness of this approach is that authorized users must be told the encryption key, and the mechanism for communicating this information is vulnerable to clever intruders.

#### Public Key Encryption

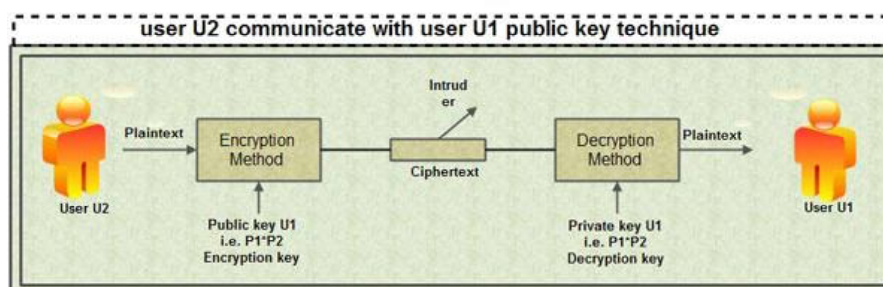
Another approach to encryption, called public-key encryption, has become increasingly popular in recent years. The encryption scheme proposed by Rivest, Shamir, and Adheman, called RSA, is a well-known example of public-key encryption. Each authorized user has a public encryption key, known to everyone and a private decryption key (used by the decryption algorithm), chosen by the user and known only to him or her. The encryption and decryption algorithms themselves are assumed to be publicly known.

Consider user called Suneet. Anyone can send Suneet a secret message by encrypting the message using Suneet's publicly known encryption key. Only Suneet can decrypt this secret message because the decryption algorithm required Suneet's decryption key, known only to Suneet. Since users choose their own decryption keys, the weakness of DES is avoided.

The main issue for public-key encryption is how encryption and decryption keys are chosen. Technically, public-key encryption algorithms rely on the existence of one-way functions, which are functions whose inverse is computationally very hard to determine.

The RSA algorithm, for example is based on the observation that although checking whether a given number of prime is easy, determining the prime factors of a nonprime number is extremely hard. (Determining the prime factors of a number with over 100 digits can take years of CPU-time on the fastest available computers today.)

We now sketch the intuition behind the RSA algorithm, assuming that the data to be encrypted is an integer 1. To choose an encryption key and a decryption key, our friend Suneet— create a public key by computing the product of two large prime numbers:  $P1$  and  $P2$ . The private key consists of the pair  $(P1, P2)$  and decryption algorithms cannot be used if the product of  $P1$  and  $P2$  is known. So we publish the product  $P1 * P2$ , but an unauthorized user would need to be able to factor  $P1P2$  to steal data. By choosing  $P1$  and  $P2$  to be sufficiently large (over 100 digits), we can make it very difficult (or nearly impossible) for an intruder to factorize it.



Although this technique is secure, but it is also computationally expensive. A hybrid scheme used for secure communication is to use DES keys exchanged via a public-key encryption scheme and DES encryption is used on the data transmitted subsequently.

### 4.13 REDUNDANT ARRAY OF INDEPENDENT DISKS (RAID)

**Q30. Explain the concept of RAID.**

*Ans. :*

**(Imp.)**

RAID or Redundant Array of Independent Disks, is a technology to connect multiple secondary storage devices and use them as a single storage media. RAID consists of an array of disks in which multiple disks are connected together to achieve different goals. RAID levels define the use of disk arrays.

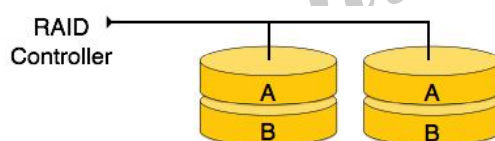
#### RAID 0

In this level, a striped array of disks is implemented. The data is broken down into blocks and the blocks are distributed among disks. Each disk receives a block of data to write/read in parallel. It enhances the speed and performance of the storage device. There is no parity and backup in Level 0.



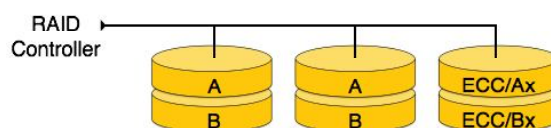
#### RAID 1

RAID 1 uses mirroring techniques. When data is sent to a RAID controller, it sends a copy of data to all the disks in the array. RAID level 1 is also called mirroring and provides 100% redundancy in case of a failure.



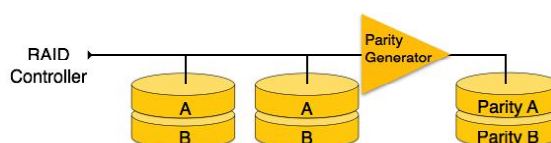
#### RAID 2

RAID 2 records Error Correction Code using Hamming distance for its data, striped on different disks. Like level 0, each data bit in a word is recorded on a separate disk and ECC codes of the data words are stored on a different set disks. Due to its complex structure and high cost, RAID 2 is not commercially available.



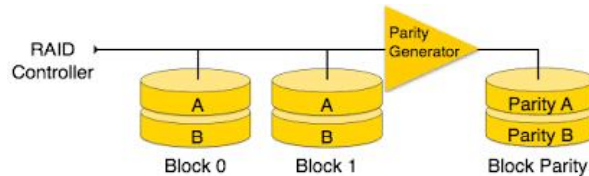
#### RAID 3

RAID 3 stripes the data onto multiple disks. The parity bit generated for data word is stored on a different disk. This technique makes it to overcome single disk failures.

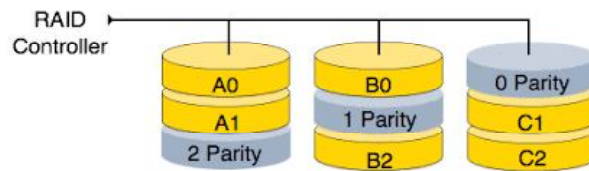


**RAID 4**

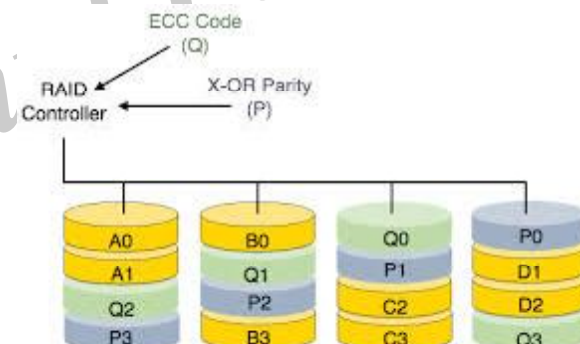
In this level, an entire block of data is written onto data disks and then the parity is generated and stored on a different disk. Note that level 3 uses byte-level striping, whereas level 4 uses block-level striping. Both level 3 and level 4 require at least three disks to implement RAID.

**RAID 5**

RAID 5 writes whole data blocks onto different disks, but the parity bits generated for data block stripe are distributed among all the data disks rather than storing them on a different dedicated disk.

**RAID 6**

RAID 6 is an extension of level 5. In this level, two independent parities are generated and stored in distributed fashion among multiple disks. Two parities provide additional fault tolerance. This level requires at least four disk drives to implement RAID.



## Short Question and Answers

### 1. What is Transaction in DBMS?

*Ans :*

A transaction is a set of changes that must all be made together. It is a program unit whose execution may or may not change the contents of a database. Transaction is executed as a single unit. If the database was in consistent state before a transaction, then after execution of the transaction also, the database must be in a consistent state. For example, a transfer of money from one bank account to another requires two changes to the database both must succeed or fail together.

You are working on a system for a bank. A customer goes to the ATM and instructs it to transfer Rs. 1000 from savings to a checking account. This simple transaction requires two steps:

- Subtracting the money from the savings account balance.
- Adding the money to the checking account balance.

### 2. Concurrency control

*Ans :*

- i) Transaction management (TM) handles all transactions properly in DBMS. Database transactions are the events or activities such as series of data read/write operations on data object(s) stored in database system.
- ii) Concurrency control (CC) is a process to ensure that data is updated correctly and appropriately when multiple transactions are concurrently executed in DBMS.

### 3. Why do we need Concurrency control

*Ans :*

In general, concurrency control is an essential part of TM. It is a mechanism for correctness when two or more database transactions that access the same data or data set are executed concurrently with time overlap. According to Wikipedia.org, if multiple transactions are executed serially or

sequentially, data is consistent in a database. However, if concurrent transactions with interleaving operations are executed, some unexpected data and inconsistent result may occur. Data interference is usually caused by a write operation among transactions on the same set of data in DBMS.

There are two main kinds of concurrency control mechanisms :

- **Pessimistic (conservative) concurrency control:** The pessimistic concurrency control delays the transactions if they conflict with other transactions at some time in the future by locking or a time-stamping technique.
- **Optimistic concurrency control:** According to Kung and Robinson (1981), the optimistic concurrency control, that assumes that the conflict is rare, allows concurrent transactions to proceed without imposing delays to ensure serializability then check conflict only at the end, when a transaction commits. Notice that there is another mechanism, semi-optimistic technique, which uses lock operations in some situations (if they may violate some rules), and does not lock in other circumstances.
- **The pros and cons of the pessimistic and optimistic concurrency control mechanisms:** Both pessimistic and optimistic concurrency control mechanisms provide different performance, e.g., the different average transaction completion rates or throughput, depending on transaction types mix, computing level of parallelism, and other events.

### 4. Define Serializability. State the importance of Serializability?

*Ans :*

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.



- **Schedule:** A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
- **Serial Schedule:** It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

In a multi-transaction environment, serial schedules are considered as a benchmark. The execution sequence of an instruction in a transaction cannot be changed, but two transactions can have their instructions executed in a random fashion. This execution does no harm if two transactions are mutually independent and working on different segments of data; but in case these two transactions are working on the same data, then the results may vary. This ever-varying result may bring the database to an inconsistent state.

To resolve this problem, we allow parallel execution of a transaction schedule, if its transactions are either Serializable or have some equivalence relation among them.

## 5. Recoverability

*Ans :*

A transaction may not execute completely due to hardware failure, system crash or software issues. In that case, we have to rollback the failed transaction. But some other transaction may also have used values produced by failed transaction. So we have to rollback those transactions as well.

## 6. What is Lock.

*Ans :*

Transaction processing systems usually allow multiple transactions to run concurrently. By allowing multiple transactions to run concurrently will improve the performance of the system in terms of increased throughput or improved response time, but this allows causes several complications with consistency of the data. Ensuring consistency in spite of

concurrent execution of transaction require extra work, which is performed by the concurrency controller system of DBMS.

A lock is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it. Generally, there is one lock for each dataitem in the database. Locks are used as a means of synchronizing the access by concurrent transactions to the database item.

## 7. Types of Locks

*Ans :*

### 1. Binary Locks

A binary lock can have two states or values: locked and unlocked.

A distinct lock is associated with each database item A. If the value of the lock on A is 1, item A cannot be accessed by a database operation that requests the item. If the value of the lock on A is 0 then item can be accessed when requested. We refer to the current value of the lock associated with item A as LOCK (A). There are two operations, lock item and unlock item are used with binary locking A transaction requests access to an item A by first issuing a lockitem (A) operation. If LOCK (A) = 1, the transaction is forced to wait. If LOCK (A) = 0 it is set to 1 (the transaction locks the item) and the transaction is allowed to access item A. When the transaction is through using the item, it issues an unlock item (A) operation, which sets LOCK (A) to 0 (unlocks the item) so that A may be accessed by other transactions. Hence binary lock enforces mutual exclusion on the data item.

### Rules of Binary Locks

If the simple binary locking scheme described here is used, every transaction must obey the following rules :

1. A transaction must issue the operation lock\_item (A) before any read\_item (A) or write\_item (A) operations are performed in T.
2. A transaction T must issue the operation unlock\_item (A) after all read\_item (A) and write\_item (A) operations are completed in T.

4. A transaction T will not issue a lock\_item (A) operation if it already holds the lock on Item A.
5. A transaction T will not issue an unlock\_item (A) operation unless it already holds the lock on item A.
6. The lock manager module of the DBMS can enforce these rules. Between the Lock\_item (A) and unlock\_item (A) operations in transaction T, is said to hold the lock on item A. At most one transaction can hold the lock on a particular item. Thus no two transactions can access the same item concurrently.

### Disadvantages of Binary Locks

#### i) Share/Exclusive (for Read/Write) Locks

We should allow several transactions to access the same item A if they all access A for reading purposes only. However, if a transaction is to write an item A, it must have exclusive access to A. For this purpose, a different type of lock called a multiple-mode lock is used. In this scheme there are shared/exclusive or read/write locks are used.

#### ii) Locking Operations

There are three locking operations called read\_lock(A), write\_lock(A) and unlock(A) represented as lock-S(A), lock-X(A), unlock(A) (Here, S indicates shared lock, X indicates exclusive lock) can be performed on a data item. A lock associated with an item A, LOCK (A), now has three possible states: "read-locked", "write-locked," or "unlocked." A read-locked item is also called share-locked item because other transactions are allowed to read the item, whereas a write-locked item is caused exclusive-locked, because a single transaction exclusively holds the lock on the item.

### 2. Compatibility of Locks

Suppose that there are A and B two different locking modes. If a transaction T1 requests a lock of mode on item Q on which transaction T2 currently hold a lock of mode B. If transaction can be granted lock, in spite of the presence of the mode B lock, then we say mode A is compatible with mode B. Such a function is shown in one matrix as shown below :

	S	X
S	true	false
X	false	false

**Compatibility Graph**

The graphs shows that if two transactions only read the same data object they do not conflict, but if one transaction writes a data object and another either read or write the same data object, then they conflict with each other. A transaction requests a shared lock on data item Q by executing the lock-S(Q) instruction. Similarly, an exclusive lock is requested through the lock- X(Q) instruction. A data item Q can be unlocked via the unlock(Q) instruction.

To access a data item, transaction T1 must first lock that item. If the data item is already locked by another transaction in an incompatible mode, the concurrency control manager will not grant the lock until all incompatible locks held by other transactions have been released. Thus, T1 is made to wait until all incompatible locks held by other transactions have been released.

**8. Define deadlock.**

*Ans :*

In a multi-process system, deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.

For example, assume a set of transactions  $\{T_0, T_1, T_2, \dots, T_n\}$ .  $T_0$  needs a resource X to complete its task. Resource X is held by  $T_1$ , and  $T_1$  is waiting for a resource Y, which is held by  $T_2$ .  $T_2$  is waiting for resource Z, which is held by  $T_0$ . Thus, all the processes wait for each other to release resources. In this situation, none of the processes can finish their task. This situation is known as a deadlock.

---

**9. Deadlock Prevention**

*Ans :*

Deadlock Prevention ensures that the system never enters a deadlock state.

Following are the requirements to free the deadlock :

- **No Mutual Exclusion** : No Mutual Exclusion means removing all the resources that are sharable.
  - **No Hold and Wait** : Removing hold and wait condition can be done if a process acquires all the resources that are needed before starting out.
  - **Allow Preemption** : Allowing preemption is as good as removing mutual exclusion. The only need is to restore the state of the resource for the preempted process rather than letting it in at the same time as the preemptor.
  - **Removing Circular Wait** : The circular wait can be removed only if the resources are maintained in a hierarchy and process can hold the resources in increasing the order of precedence.
- 

**10. What is Database Recovery ?**

*Ans :*

Data recovery is the art of restoring lost or damaged files. This damage can occur when your computer crashes, a virus infects, you accidentally reformat a disk that contains precious data, or you experience some other catastrophe of considerable dimension. And, at some point in your life, you're going to delete a file you really didn't mean to (believe me). The next time tragedy strikes, try running one of the many data recovery applications (powerful software written specifically for data recovery purposes) to see if it can correct the situation. Often these little jewels work magic and save your day-and your files.

There are companies that specialize in restoring lost data. These people generally work with more serious situations, like when you lose the last three years of your business's financial records. Find them in the yellow pages under "Computers: Service and Repair." Check the ads in the back pages of computer magazines if you can't find someone local. It's very common to have to ship your hard disk or your computer off to a company that can fix it.

---

**11. Data Security?**

*Ans :*

Threats and risks to databases have increased and therefore, the need for securing databases has also increased. Let's learn about the basic facets of database security, including assurance, integrity, availability, and confidentiality.

The majority of the companies store sensitive data in databases. However, database security is sometimes not given as much thought and effort as other areas of computer security. Hackers have been able to target large databases in recent years to obtain sensitive information like credit card numbers and other personal information. It is important to protect databases against these risks, and this is where database security comes into place.

Database security can be defined as a system or process by which the "Confidentiality, Integrity, and Availability," or CIA, of the database can be protected. Unauthorized entry or access to a database server signifies a loss of confidentiality; unauthorized alteration to the available data signifies loss of integrity; and lack of access to database services signifies loss of availability. Loss of one or more of these basic facets will have a significant impact on the security of the database.

For an illustration of this concept, imagine that the website of a company contains information like who they are, what they do, and what prospective customers have to do to contact them for their queries. In this case, the availability of the database services is more important when compared with other factors like the confidentiality or integrity of the database security.

---

## **12. Authorization.**

*Ans :*

Authorization is a security mechanism used to determine user/client privileges or access levels related to system resources, including computer programs, files, services, data and application features. Authorization is normally preceded by authentication for user identity verification. System administrators (SA) are typically assigned permission levels covering all system and user resources.

During authorization, a system verifies an authenticated user's access rules and either grants or refuses resource access.

Modern and multiuser operating systems depend on effectively designed authorization processes to facilitate application deployment and management. Key factors include user type, number, and credentials requiring verification and related actions and roles. For example, role-based authorization may be designated by user groups requiring specific user resource tracking privileges. Additionally, authorization may be based on an enterprise authentication mechanism, like Active Directory (AD), for seamless security policy integration.

---

## **13. Explain the concept of Database Encryption and Decryption.**

*Ans :*

Database encryption is the process of converting data, within a database, in plain text format into a meaningless cipher text by means of a suitable algorithm.

Database decryption is converting the meaningless cipher text into the original information using keys generated by the encryption algorithms.

Database encryption can be provided at the file or column level.

Encryption of a database is costly and requires more storage space than the original data. The steps in encrypting a database are :

1. Determine the criticality of the need for encryption
2. Determine what data needs to be encrypted
3. Determine which algorithms best suit the encryption standard
4. Determine how the keys will be managed.

Numerous algorithms are used for encryption. These algorithms generate keys related to the encrypted data. These keys set a link between the encryption and decryption procedures. The encrypted data can be decrypted only by using these keys.

#### 14. RAID

*Ans :*

RAID or Redundant Array of Independent Disks, is a technology to connect multiple secondary storage devices and use them as a single storage media. RAID consists of an array of disks in which multiple disks are connected together to achieve different goals. RAID levels define the use of disk arrays.

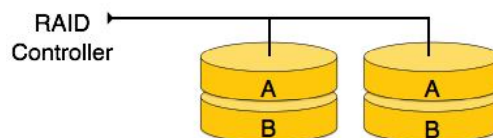
##### RAID 0

In this level, a striped array of disks is implemented. The data is broken down into blocks and the blocks are distributed among disks. Each disk receives a block of data to write/read in parallel. It enhances the speed and performance of the storage device. There is no parity and backup in Level 0.



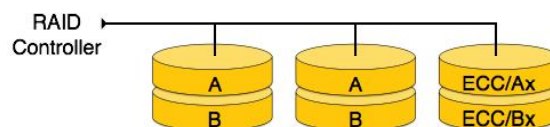
##### RAID 1

RAID 1 uses mirroring techniques. When data is sent to a RAID controller, it sends a copy of data to all the disks in the array. RAID level 1 is also called mirroring and provides 100% redundancy in case of a failure.



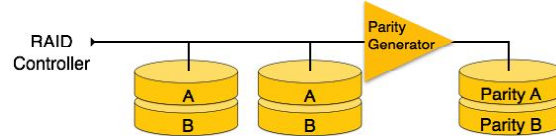
##### RAID 2

RAID 2 records Error Correction Code using Hamming distance for its data, striped on different disks. Like level 0, each data bit in a word is recorded on a separate disk and ECC codes of the data words are stored on a different set disks. Due to its complex structure and high cost, RAID 2 is not commercially available.

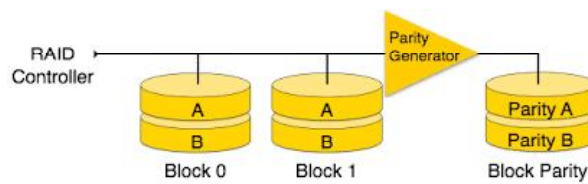


**RAID 3**

RAID 3 stripes the data onto multiple disks. The parity bit generated for data word is stored on a different disk. This technique makes it to overcome single disk failures.

**RAID 4**

In this level, an entire block of data is written onto data disks and then the parity is generated and stored on a different disk. Note that level 3 uses byte-level striping, whereas level 4 uses block-level striping. Both level 3 and level 4 require at least three disks to implement RAID.



## Choose the Correct Answers

1. Identify the characteristics of transactions. [ d ]  
(a) Atomicity (b) Durability  
(c) Isolation (d) All
2. RDBMS uses \_\_\_\_\_ statement to declare a new transaction to start and its properties [ b ]  
(a) Begin (b) Set transaction  
(c) Begin transaction (d) commit
3. Detecting system failure and restoring database is known as \_\_\_\_\_ [ a ]  
(a) failure recovery (b) failure identification  
(c) failure response (d) failure management
4. Which of the following are introduced to reduce the overhead caused by the log-based recovery is \_\_\_\_\_ [ d ]  
(a) checkpoints (b) indices  
(c) dead locks (d) locks
5. Which of the following protocols ensures conflict serialisability in deadlocks is \_\_\_\_\_ [ b ]  
(a) two phase locking protocol (b) time stamp ordering protocol  
(c) graph based protocol (d) none
6. \_\_\_\_\_ is used for database security [ d ]  
(a) data encryption (b) a view  
(c) finger print (d) All
7. Prevention of access to the database by unauthorised user is \_\_\_\_\_ [ c ]  
(a) integrity (b) productivity  
(c) security (d) reliability
8. Authentication refers to \_\_\_\_\_ [ d ]  
(a) Restricting user (b) Controlling access to database  
(c) controlling operations (d) All
9. What are the common security threats \_\_\_\_\_ [ b ]  
(a) File shredding (b) File sharing permissions  
(c) File corruption (d) File integrity
10. Which level of RAID refers to disk mirroring with block striping. [ a ]  
(a) RAID level 1 (b) RAID level 2  
(c) RAID level 0 (d) RAID level 3

## *Fill in the Blanks*

1. \_\_\_\_\_ is a set of changes that must be made together.
2. Concurrency control is a process to \_\_\_\_\_ and \_\_\_\_\_ when multiple transactions occur.
3. A transaction database system must maintain \_\_\_\_\_
4. A concurrency control program is called \_\_\_\_\_
5. A \_\_\_\_\_ is a condition in which 2 or more conditions are waiting.
6. A dead lock is called as \_\_\_\_\_ condition
7. \_\_\_\_\_ is a record of last transaction accessed using time stamp
8. \_\_\_\_\_ is a phenomenon of uncommitted data occurring when 2 transactions exist
9. \_\_\_\_\_ is known as executing multiple transactions
10. Two schedules are said to be \_\_\_\_\_ when one transaction transforms into other.
11. A \_\_\_\_\_ is a variable associated with data item and its status.
12. \_\_\_\_\_ ensure that DBMS never enter into dead lock state.
13. \_\_\_\_\_ is a process of removing the transactions.
14. \_\_\_\_\_ is a process of restoring lost or damaged files.
15. Recovery techniques are heavily dependent upon the special file known as \_\_\_\_\_

### ANSWERS

1. Transaction
2. Update correctly, appropriate
3. ACID Properties
4. Granularity
5. Dead lock
6. Circular waiting
7. Granular Time Stamp
8. Isolation property
9. Serializability
10. Conflict equivalent
11. Lock
12. Dead lock prevention
13. Termination
14. Data recover
15. System log



FACULTY OF SCIENCE  
**B.Sc. IV-Semester (CBCS) Examination**  
**Model Paper - I**  
**DATABASE MANAGEMENT SYSTEMS**

Time : 3 Hours]

[Max. Marks : 80

**PART- A (8 × 4 = 32 M)**

**[Short Answer Type]**

**Note :** Answer any Eight of the following questions

- |  |                    |
|--|--------------------|
| 1. What is the purpose of DBMS?          | (Unit-I, SQA-3)    |
| 2. Who is database administrator.        | (Unit-I, SQA-12)   |
| 3. Data Manipulation Language (DML).     | (Unit-I, SQA-5)    |
| 4. What is Data Redundancy?              | (Unit-II, SQA-3)   |
| 5. Normalization                         | (Unit-II, SQA-9)   |
| 6. List out the benefits of ER Diagrams. | (Unit-II, SQA-5)   |
| 7. What is the use of Null values?       | (Unit-III, SQA-4)  |
| 8. What are the advantages of SQL?       | (Unit-III, SQA-2)  |
| 9. What is PL/SQL Trigger ?              | (Unit-III, SQA-14) |
| 10. Concurrency control                  | (Unit-IV, SQA-2)   |
| 11. Define deadlock.                     | (Unit-IV, SQA-8)   |
| 12. Authorization                        | (Unit-IV, SQA-12)  |

**SECTION - B (4 × 12 = 48 M)**

**[Essay Answer Type]**

**Note :** Answer all the following questions

- |  |                             |
|--|-----------------------------|
| 13. (a) List out various applications of database.   | (Unit-I, Q.No. 5)           |
| (OR)   |                             |
| (b) Explain the major functional components of database.   | (Unit-I, Q.No. 20)          |
| 14. (a) What is relational integrity constraints in DBMS ?   | (Unit-II, Q.No. 8)          |
| (OR)   |                             |
| (b) Explain in detail 1NF, 2NF, 3NF  | (Unit-II, Q.No. 34, 38, 39) |
| 15. (a) What is mean by view in SQL? Explain the process of creating, updating and dropping a view ? | (Unit-III, Q.No. 29)        |
| (OR)   |                             |
| (b) Describe in detail about DDL commands in SQL ?   | (Unit-III, Q.No. 7)         |
| 16. (a) What is Lock? Explain various type of Locks in DBMS.   | (Unit-IV, Q.No. 9)          |
| (OR)   |                             |
| (b) Explain the working mechanism of recovery.   | (Unit-IV, Q.No. 15)         |

FACULTY OF SCIENCE  
**B.Sc. IV-Semester (CBCS) Examination**  
**Model Paper - II**  
**DATABASE MANAGEMENT SYSTEMS**

Time : 3 Hours]

[Max. Marks : 80

**PART- A (8 × 4 = 32 M)****[Short Answer Type]****Note :** Answer any Eight of the following questions

- |   |                    |
|---|--------------------|
| 1. Define the term transaction.                                     | (Unit-I, SQA-10)   |
| 2. What is relational model ?                                       | (Unit-I, SQA-13)   |
| 3. What is Database?  | (Unit-I, SQA-1)    |
| 4. Explain the concept of Generalization.                           | (Unit-II, SQA-6)   |
| 5. What is database design?   | (Unit-II, SQA-1)   |
| 6. Advantages of Normalization.                                     | (Unit-II, SQA-17)  |
| 7. What are the differences between grant and revoke.               | (Unit-III, SQA-3)  |
| 8. What is Right Join?  | (Unit-III, SQA-8)  |
| 9. What is function ? Write the syntax to declare functions in SQL. | (Unit-III, SQA-12) |
| 10. Why do we need Concurrency control.                             | (Unit-IV, SQA-3)   |
| 11. Types of Locks  | (Unit-IV, SQA-7)   |
| 12. Data Security?  | (Unit-IV, SQA-11)  |

**SECTION - B (4 × 12 = 48 M)****[Essay Answer Type]****Note :** Answer all the following questions

- |  |                      |
|--|----------------------|
| 13. (a) Explain ACID Properties and how they are useful to transactions?<br>(OR)       | (Unit-I, Q.No. 22)   |
| (b) Explain in detail various types of integrity constraints?                          | (Unit-I, Q.No. 31)   |
| 14. (a) Explain the process of creating ER Diagram into table with an example.<br>(OR) | (Unit-II, Q.No. 18)  |
| (b) Explain database designing process ?   | (Unit-II, Q.No. 4)   |
| 15. (a) List out various types of SQL Commands.<br>(OR)                                | (Unit-III, Q.No. 6)  |
| (b) What is Join? List out various types of joins supported by SQL?                    | (Unit-III, Q.No. 23) |
| 16. (a) State the architecture of transaction management.<br>(OR)                      | (Unit-IV, Q.No. 3)   |
| (b) Explain different Recovery Techniques/Approaches in DBMS ?                         | (Unit-IV, Q.No. 18)  |

FACULTY OF SCIENCE  
**B.Sc. IV-Semester (CBCS) Examination**  
**Model Paper - III**  
DATABASE MANAGEMENT SYSTEMS

Time : 3 Hours]

[Max. Marks : 80

**PART- A (8 × 4 = 32 M)****[Short Answer Type]****Note :** Answer any Eight of the following questions

- |   |                    |
|---|--------------------|
| 1. Define DBMS.   | (Unit-I, SQA-2)    |
| 2. Data Control Language  | (Unit-I, SQA-6)    |
| 3. List out various types of Keys.                              | (Unit-I, SQA-14)   |
| 4. Define Decomposition ?                                       | (Unit-II, SQA-13)  |
| 5. What is Functional Dependency ?                              | (Unit-II, SQA-12)  |
| 6. What is ER Model ?   | (Unit-II, SQA-2)   |
| 7. What is SQL ?  | (Unit-III, SQA-1)  |
| 8. What is transaction? Explain the properties of transaction ? | (Unit-III, SQA-10) |
| 9. Nested subquery  | (Unit-III, SQA-5)  |
| 10. Authorization.  | (Unit-IV, SQA-12)  |
| 11. Deadlock Prevention   | (Unit-IV, SQA-9)   |
| 12. What is Lock.   | (Unit-IV, SQA-6)   |

**SECTION - B (4 × 12 = 48 M)****[Essay Answer Type]****Note :** Answer all the following questions

- |  |                      |
|--|----------------------|
| 13. (a) What is relational algebra? List and explain various types of relational operations. | (Unit-I, Q.No. 43)   |
| (OR)   |                      |
| (b) Explain various techniques used to view data in database.                                | (Unit-I, Q.No. 11)   |
| 14. (a) List out advantages and disadvantages of data redundancy.                            | (Unit-II, Q.No. 11)  |
| (OR)   |                      |
| (b) What is meant by Normalization ?   | (Unit-II, Q.No. 31)  |
| 15. (a) Explain the use of as clause in SQL.   | (Unit-III, Q.No. 15) |
| (OR)   |                      |
| (b) List and explain basic data types supported by SQL ?                                     | (Unit-III, Q.No. 4)  |
| 16. (a) Discuss in detail about the Recoverability of Schedules ?                            | (Unit-IV, Q.No. 8)   |
| (OR)   |                      |
| (b) Explain the Concepts behind the Data Security?   | (Unit-IV, Q.No. 20)  |