KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2018 onwards)

DEPARTMENT OF COMPUTER SCIENCE, COMPUTER APPLICATION & INFORMATION TECHNOLOGY

SUBJECT: RELATIONAL DATABASE MANAGEMENT SYSTEM
SEMESTER:III
SUBJECT CODE: 18CTU303 **CLASS: II B.Sc. CT**

S.No	Lecture Duration (Period)	Topics to be Covered	Support Materials
Unit – I			
1	1	DBMS Definition, Characteristics of DBMS	S1:2,W4
2	1	Application and advantages of DBMS, Instances	S1:27-29,W4
3	1	Schemas and Database States	S1:14-15,W4
4	1	Three Levels of Architecture	S1:15-16,W4
5	1	Data Independence, DBMS languages	S1:17-19,W4
6	1	Data Dictionary	S1:26-27,W4
7	1	Database Users	W4
8	1	Data Administrators	S1:21 W4
9	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-I	**09**
Unit – II			
1	1	Data Models, types and their comparison	S1:34,35,W4
2	1	Entity Relationship Model, Entity Types	W4
3	1	Entity Sets, Attributes and its types	S1:2-7,36-41,W4
4	1	Keys	S1:159-161,W4
5	1	E-R Diagram	S4:80-81,W4
6	1	Data Integrity RDBMS –Concept Components	S1:21-22,W1
7	1	Codd's rules	S4:179-180,W1
8	1	Codd's rules Continued	S4:179-180,W1
9	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-II	**09**

Unit – III			
1	1	Relational Algebra selection, projection	S1:165-166,W4
2	1	union, intersection, Cartesian product	S1:167-169,W4
3	1	Different types of join like theta join equi-join, natural join, outer join	S4:156-157,W4
4	1	Functional Dependencies and Good & Bad Decomposition	S4:317-322,W4
5	1	Anomalies as a database: A consequences of bad design	S4:337,W4
6	1	Normalization: 1NF, 2NF,	S4:640-346,W4
7	1	3NF, BCNF,	S4:347-351,W4
8	1	4NF 5NF	S4:352-360,W4
9	1	Recapitulation and Discussion of important questions	
Total No. of Hours Planned for Unit-III			09
Unit – IV			
1	1	Introduction to SQL: DDL, DML, and DCL statements	S1:209,W1,W2
2	1	Creating Tables, Adding Constraints, Altering Tables	S1:212-219,W1,W2
3	1	Update, Insert, Delete	W1,W2
4	1	various Form of SELECT	S1:217-216,W1,W2
5	1	Simple, Using Special Operators for Data Access	W1,W2
6	1	Aggregate functions, Joining Multiple Tables (Equi Joins),Joining a Table to itself (self Joins) Functions	W1,W2
7	1	Introduction to PL/SQL : Declaration section executable command section	S6: 490-496, W3
8	1	conditional logic loops CASE statements	S6: 496-502, W3
9	1	Recapitulation and Discussion of important questions	
Total No. of Hours Planned for Unit-IV			**09**

Unit – V			
1	1	Exception handling section: predefined and user defined exceptions	S6: 503-506,W3
2	1	Triggers: definition – types: row level, statement level before and after, instead of – syntax	S6: 511-514
3	1	Trigger: enabling and disabling triggers - replacing and dropping triggers	S6: 511-514
4	1	Cursors – definition – open – fetch – close	S1: 189-191,W3
5	1	Cursor attributes- select for update implicit, explicit	S1: 189-191,W3
6	1	Procedures, Functions: Local and global procedures vs functions – stored procedures functions – create procedure syntax-	W3
7	1	Create function syntax – calling procedures, functions. Replacing and dropping procedures, functions	W3
8	1	Package header – package body calling package members - Replacing and dropping package	W3
9	1	Recapitulation and Discussion of important questions	
10	1	Pervious ESE Question Paper Discussion	
11	1	Pervious ESE Question Paper Discussion	
12	1	Pervious ESE Question Paper Discussion	
		Total No. of Hours Planned for Unit-V	**12**
		Total No Hours	**48**

Suggested Readings

S1. Bipin C. Desai. (2013). An Introduction to Database Systems, New Delhi: Galgotia Publications.

S4. Shio Kumar Singh (2011). Database Management Systems – Concepts, design and Applications (2nd ed.). New Delhi: Pearson Education.

S6: Kevin Loney and George Koch. 2002. Oracle 9i The Complete Reference, 1st Edition, Tata Mcgraw-Hill, New Delhi

Websites

W1. http://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm

W2. www.databasedir.com

W3. http://plsql-tutorial.com/

W4. https://www.javatpoint.com/dbms-tutorial

Faculty **HoD**

RDBMS (CT) - III Sem.

Unit I

DBMS Definition, Characteristics of DBMS ,Application and advantages of DBMS, Instances , Schemas and Database States, Three Levels of Architecture , Data Independence, DBMS languages, Data Dictionary, Database Users, Data Administrators.

Unit II

Data Models, types and their comparison, Entity Relationship Model, Entity Types, Entity Sets, Attributes and its types, Keys, E-R Diagram, Data Integrity RDBMS –Concept, Components and Codd's rules.

Unit III

Relational Algebra (selection, projection, union, intersection, Cartesian product, Different types of join like theta join, equi-join, natural join, outer join)

Functional Dependencies, Good & Bad Decomposition, Anomalies as a database: A consequences of bad design, Normalization: 1NF, 2NF, 3NF, BCNF, 4NF 5NF.

Unit IV

Introduction to SQL, DDL, DML, and DCL statements, Creating Tables, Adding Constraints, Altering Tables, Update, Insert, Delete & various Form of SELECT- Simple, Using Special Operators for Data Access. Aggregate functions, Joining Multiple Tables (Equi Joins),Joining a Table to itself (self Joins) Functions.

Introduction to PL/SQL: Declaration section – executable command section : conditional logic, loops, CASE statements –

Unit V

Exception handling section: predefined and user defined exceptions. Triggers: definition – types: row level, statement level, before and after, instead of – syntax – enabling and disabling triggers - replacing and dropping triggers. Cursors – definition – open – fetch – close – cursor attributes-select for update – types : implicit, explicit. Procedures, Functions: Local and global – procedures vs functions – stored procedures, functions – create procedure syntax - create function syntax – calling procedures, functions. Replacing and dropping procedures, functions. Package header – package body – calling package members - Replacing and dropping package.

Suggested Readings

1. Bipin C. Desai.(2013). An Introduction to Database Systems, New Delhi: Galgotia Publications.
2. Rajiv chopra (2013). Database Management systems (3rd ed.). S.Chand publications.
3. Steven Feurstein, Bill Pribyl (2014). Oracle PL/SQL Programming (6th ed.). O ' Reilly Media.
4. Shio Kumar Singh (2011). Database Management Systems – Concepts, design and Applications (2nd ed.). New Delhi: Pearson Education.
5. Ivan Byross (2010). SQL, Pl/SQL the Programming Language of Oracle Paperback. BPB Publications.
6. Rajeeb C. Chatterjee (2012). Learning Oracle SQL and Pl/SQL: A simplified Guide. Prentice Hall of India.

Web Sites

1. http://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm
2. www.databasedir.com
3. http://plsql-tutorial.com/

UNIT-I

SYLLABUS

DBMS Definition– Characteristics of DBMS- Application and Advantages of DBMS- Instances-Schemas and Database States, Three Levels of Architecture - Data Independence – DBMS Languages – Data Dictionary, Database Users, Data Administrators.

DBMS Definition

A database management system, or DBMS, is software that stores, retrieves and updates files from a centralized database. It acts as an intermediary between programs and the database, and allows multiple users or programs to access a data file at once. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible.

Characteristics of DBMS

Stores any kind of data

A database management system should be able to store any kind of data. It should not be restricted to the employee name, salary and address. Any kind of data that exists in the real world can be stored in DBMS because we need to work with all kinds of data that is present around us.

Support ACID Properties

Any DBMS is able to support ACID (Accuracy, Completeness, Isolation, and Durability) properties. It is made sure is every DBMS that the real purpose of data should not be lost while performing transactions like delete, insert an update. Let us take an example; if an employee name is updated then it should make sure that there is no duplicate data and no mismatch of student information.

Represents complex relationship between data

Data stored in a database is connected with each other and a relationship is made in between data. DBMS should be able to represent the complex relationship between data to make the efficient and accurate use of data.

Backup and recovery

There are many chances of failure of whole database. At that time no one will be able to get the database back and for sure company will be in a big loss. The only solution is to take backup of database and whenever it is needed, it can be stored back. All the databases must have this characteristic.

Structures and described data

A database should not contains only the data but also all the structures and definitions of the data. This data represent itself that what actions should be taken on it. These descriptions include the structure, types and format of data and relationship between them.

Data integrity

This is one of the most important characteristics of database management system. Integrity ensures the quality and reliability of database system. It protects the unauthorized access of database and makes it more secure. It brings only the consistence and accurate data into the database.

Concurrent use of database

There are many chances that many users will be accessing the data at the same time. They may require altering the database system concurrently. At that time, DBMS supports them to concurrently use the database without any problem.

Application and Advantages of DBMS

Compared to the File Based Data Management System, Database Management System has many advantages. Some of these advantages are given below:

Reducing Data Redundancy

The file based data management systems contained multiple files that were stored in many different locations in a system or even across multiple systems. Because of this, there were sometimes multiple copies of the same file which lead to data redundancy.This is prevented in a database as there is a single database and any change in it is reflected immediately. Because of this, there is no chance of encountering duplicate data.

Sharing of Data

In a database, the users of the database can share the data among themselves. There are various levels of authorisation to access the data, and consequently the data can only be shared based on the correct authorisation protocols being followed.Many remote users can also access the database simultaneously and share the data between themselves.

Data Integrity

Data integrity means that the data is accurate and consistent in the database. Data Integrity is very important as there are multiple databases in a DBMS. All of these databases contain data that is visible to multiple users. So it is necessary to ensure that the data is correct and consistent in all the databases and for all the users.

Data Security

Data Security is vital concept in a database. Only authorised users should be allowed to access the database and their identity should be authenticated using a username and password. Unauthorised users should not be allowed to access the database under any circumstances as it violates the integrity constraints.

Privacy

The privacy rule in a database means only the authorized users can access a database according to its privacy constraints. There are levels of database access and a user can only view the data he is allowed to. For example - In social networking sites, access constraints are different for different accounts a user may want to access.

Backup and Recovery

Database Management System automatically takes care of backup and recovery. The users don't need to backup data periodically because this is taken care of by the DBMS. Moreover, it also restores the database after a crash or system failure to its previous condition.

Data Consistency

Data consistency is ensured in a database because there is no data redundancy. All data appears consistently across the database and the data is same for all the users viewing the database. Moreover, any changes made to the database are immediately reflected to all the users and there is no data inconsistency.

DBMS Instance

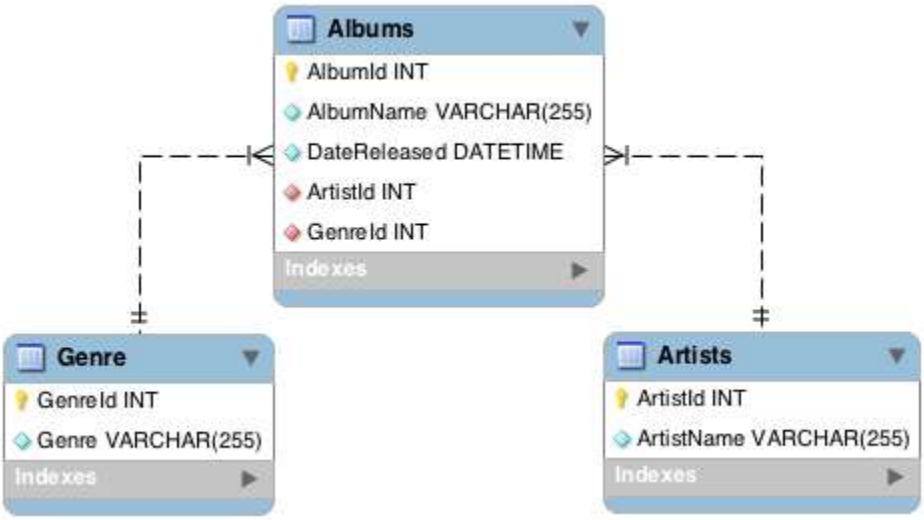
The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

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.

DBMS Schemas

- The overall design of a database is called schema.
- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
- A database schema is designed by the database designers to help programmers whose software will interact with the database. The process of database creation is called data modeling.

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 schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram.



DBMS States

Database state, in database technology the set of stored data i.e., actual data stored in a particular moment in time. Entering, modifying, or deleting information changes the database state. All these states are denoted by differences in their performance, the results produced and the ease of use. A database can operate in only one of these states at any given time. We can find out the current state of your database by selecting state_desc column present in the sys.databases catalog view.

1. **Online:** When the state of a database is Online, it means that it can be accessed at the moment, and is available. It also means that the state of Primary file group is essentially online, though the important undo phase of recovery might still be incomplete.

2. **Offline:** This simply means that the database is unavailable, this usually happens due to explicit user action, and can't be rectified till user action is taken. This can be explained with the instance of the database being offline during the transfer file to disk, and then again getting back online after completing the transfer.

3. **Restoring:** When one or more than one files of primary file group or one or more than one files of secondary files is apparently being restored, the database is in 'Restoring' state and is unavailable.

4. **Recovering:** This is a transient process; in-case of successful recovery database will be online automatically. In-case of a failure in recovery, it will become 'Suspect', and will thus be unavailable.

5. **Recovery Pending:** This means that a resource related error has been encountered during recovery. This does not damage the database, but some files may go missing or system

resource limitation may create problems while opening files. This requires additional action from the user to deal with the error and complete the recovery process. In this state also the database is unavailable.

6. **Suspect:** This marks unavailability of database for user connection and shows that database is damaged or primary filegroup is suspect, this involves action by user to solve the problem.

7. **Emergency:** This is done by the user them-self; a user can change the state of database to 'Emergency' and then it will operate on single user mode which may be repaired or even restored. The database operates as READ_ONLY, or logging is disabled, it is used primarily for the purpose of troubleshooting

Three Levels of Architecture

- The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.

1. Internal Level

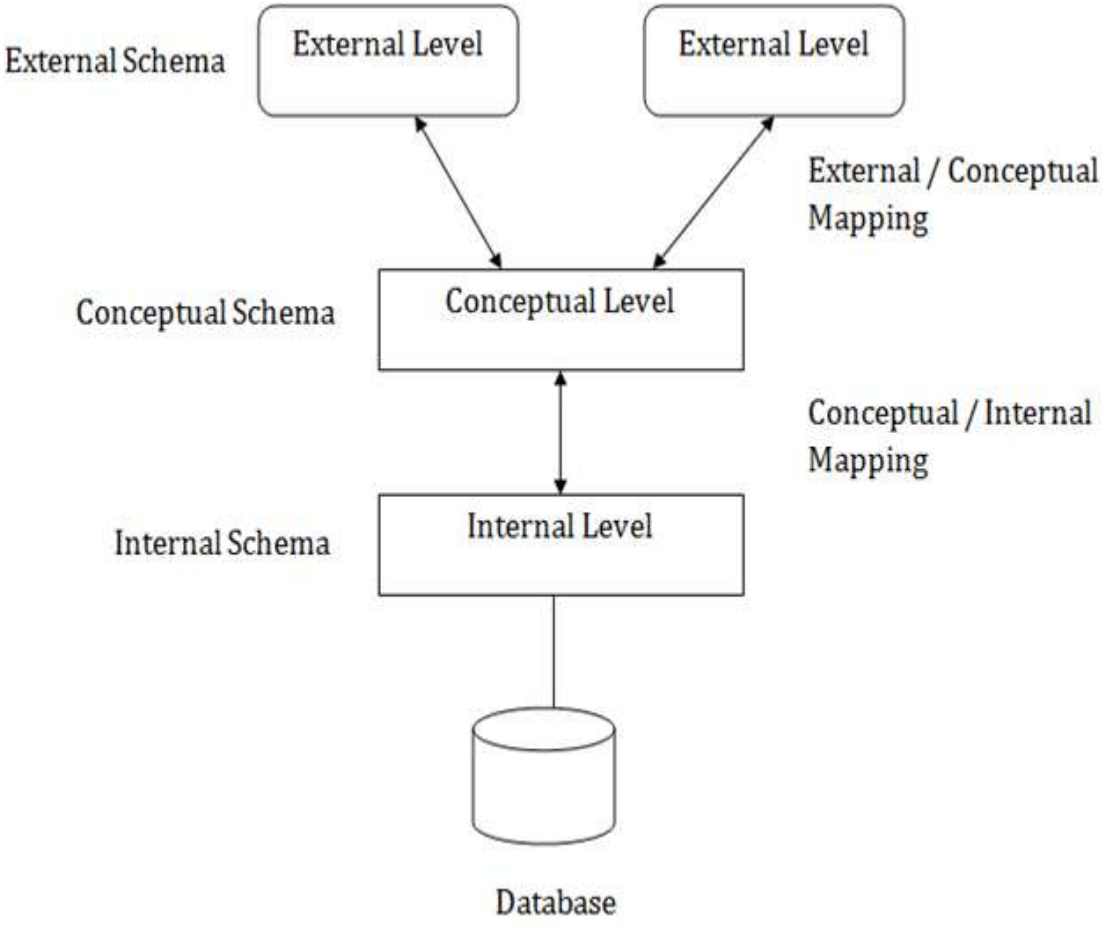
- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored in a block.
- The physical level is used to describe complex low-level data structures in detail.

2. Conceptual Level

- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.

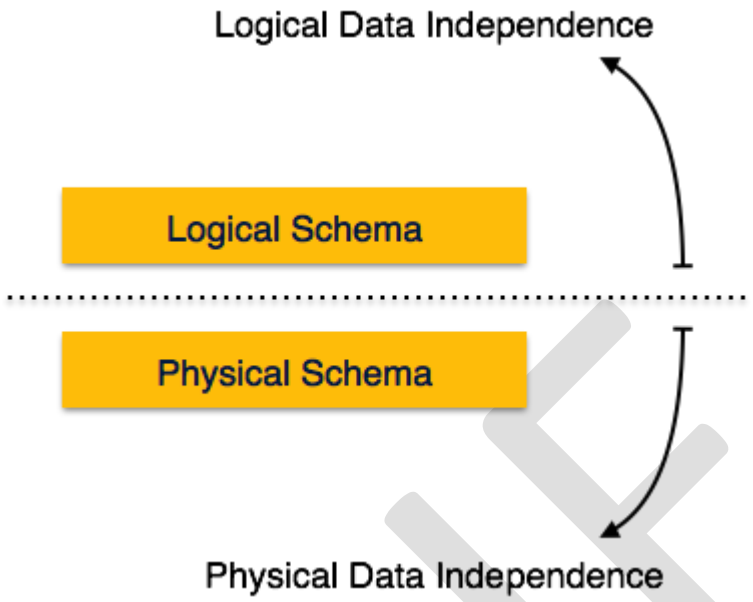
3. External Level

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.



Data Independence

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.



Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

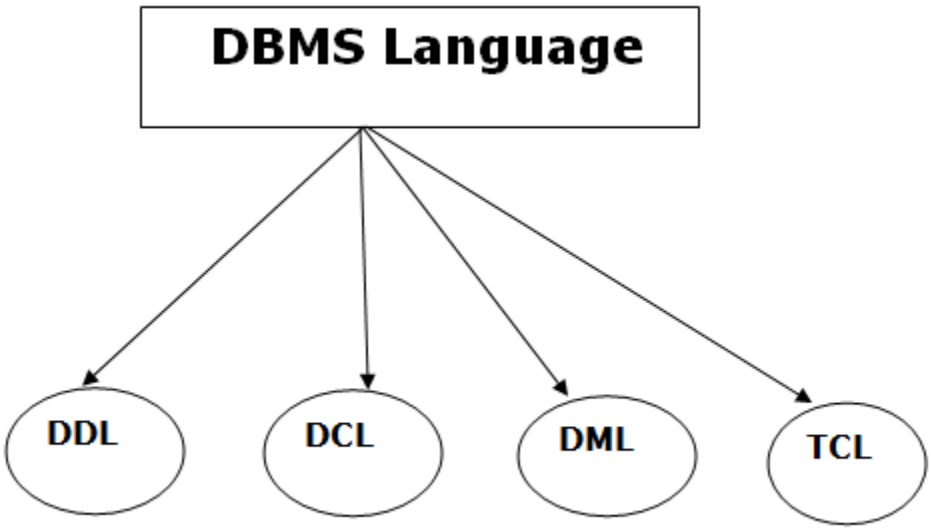
All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

DBMS Languages

- 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.

Types of Database Language



1. Data Definition Language

- **DDL** stands for **D**ata **D**efinition **L**anguage. 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 **D**ata **M**anipulation **L**anguage. 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 **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.

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.

Data Dictionary

A data dictionary contains metadata i.e., data about the database. The data dictionary is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc. The users of the database normally don't interact with the data dictionary, it is only handled by the database administrators.

The data dictionary in general contains information about the following:

1. Names of all the database tables and their schemas.
2. Details about all the tables in the database, such as their owners, their security constraints, when they were created etc.
3. Physical information about the tables such as where they are stored and how.
4. Table constraints such as primary key attributes, foreign key information etc.
5. Information about the database views that are visible.

This is a data dictionary describing a table that contains employee details.

Field Name	Data Type	Field Size for display	Description	Example
Employee Number	Integer	10	Unique ID of each employee	1645000001
Name	Text	20	Name of the employee	David Heston
Date of Birth	Date/Time	10	DOB of Employee	08/03/1995
Phone Number	Integer	10	Phone number of employee	6583648648

The different types of data dictionary are:

Active Data Dictionary

If the structure of the database or its specifications change at any point of time, it should be reflected in the data dictionary. This is the responsibility of the database management system in which the data dictionary resides.

So, the data dictionary is automatically updated by the database management system when any changes are made in the database. This is known as an active data dictionary as it is self-updating.

Passive Data Dictionary

This is not as useful or easy to handle as an active data dictionary. A passive data dictionary is maintained separately to the database whose contents are stored in the dictionary. That means that if the database is modified the database dictionary is not automatically updated as in the case of Active Data Dictionary.

So, the passive data dictionary has to be manually updated to match the database. This needs careful handling or else the database and data dictionary are out of sync.

Database Users

Database users are the one who really use and take the benefits of database. There will be different types of users depending on their need 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 particular department will involve a query to fetch the data from 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 query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.
3. **Specialized Users -** They 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 stand-alone database for their personal use. These kinds of 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.

Database Administrators

The life cycle of database starts from designing, implementing to 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 challenge. There will be unused memory in database, making the memory inevitably huge. These administration and maintenance of database is taken care by 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 comes in the market or requirement. If there is any failure in upgradation of the existing servers, he should be able revert the new changes back to the older version, thus maintaining the DBMS working. He is also responsible for updating the service packs/ hot fixes/ patches to the DBMS servers.
- **Design and implementation: -** Designing the database and implementing is also DBA's responsibility. He should be able to decide proper memory management, file organizations, error handling, log maintenance etc., for the database.
- **Performance tuning: -** Since 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 responsibility of the DBA to tune the database performance. He is responsible to make sure all the queries and programs works in 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 by 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 and security methods. He should keep various reports about database performance.

Possible Questions
Part-A
(Online – Multiple choice questions)
Included in Excel file – File name Unit-I(MCQ).xls
(Each questions carries one mark each)

PART – B (2 Marks)
1. What are attribute and its types?
2. Write a short note on DBA?
3. Write the advantages and disadvantages of relational model.
4. What is object oriented database model?
5. Write a short note on cardinality with example

PART – C (8 Marks)
1. Discuss in detail about distributed DBMS.
2. Differentiate between Hierarchical and Network database models with a neat sketch.

Write the pros and cons of each model.

3. Define a DBMS. Classify the types of users of a DBMS based on the degree their expertise.
4. Compare DBMS and DDBMS.
5. Elaborate in detail the various design concepts of a database
6. Discuss in detail about any three database models and compare them with their
7. benefits and drawbacks.
8. Enumerate in detail various types of Relationships with examples for each.
9. Enumerate in detail the major components involved in the structure of a DBMS.
10. Describe ER Medel in detail with its attributes classification.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

S.No	**Question**	**Option 1**	**Option 2**	**Option 3**	**Option 4**	**Answer**
1	_____is a Collection of data	DBMS	Database	Relationships	Entities	**Database**
2	_____is a software designed to assist in maintaining and utilizing large collections of data.	Database	DBMS	Entities	attributes.	**DBMS**
3	The data stored in database at a particular moment of time is called _____ of database.	schema	instance	model	system	**instance**
4	The overall design of a database is called	Model	Instance	Design	Schema	**Schema**
5	_____ model is also called Application interface	External	Internal	Conceptual	Logical	**External**
6	What is a SQL?	Segment Query Language	Select Query Language	Structured Query Language	Secured Query Language	**Structured Query Language**
7	_____ is used to define that how the data will be stored in a block.	View Schema	Conceptual Schema	Physical Schema	External Schema	**Physical Schema**
8	Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level is called as _____	Data Independence	Data Security	Data Integrity	Data Consistency	**Data Independence**

9	The Grant task is in _____.	DDL	DML	DCL	TCL	**DCL**
10	The task used to remove all records from a table	truncate	create	alter	drop	**truncate**
11	_____model is used to represent parent-child relationship.	Physical data model	ER model	Hierarchical model	structure chart	**ER model**
12	A structure of data in a data model is called_____	Schema	relation	record	entities	**Schema**
13	Field is otherwise known as _____	Column	Entity	Relationship	Relation	**Column**
14	Column is otherwise known as _____	Entity	Relationship	Relation	attribute	**attribute**
15	_____is used to define schemas of a table.	DDL	DML	DCL	TCL	**DDL**
16	Information about the conceptual,external and physical schemas is stored in _____	Directory	System Catalogs	IMS	Information System	**System Catalogs**
17	_____ is used to define database structure or pattern.	TCL	DML	DDL	DCL	**DDL**
18	Conceptual schema otherwise called as _____	Physical Schema	Internal Schema	Logical Schema	relations	**Logical Schema**
19	Physical Schema specifies _____details	Information	data	Storage	relationships	**Storage**
20	DBMS states are _____	Online, Offline	Submit, Run	Ready, Run	Submit, block	**Online, Offline**
21	The process of arriving at a good physical schema is called_____	Conceptual database design	Physical database design	Logical Schema	schema	**Physical database design**
22	A DBMS enable users to create, modify and query data through a _____	DDL	DML	TCL	DCL	**DML**

23	A DBMS provides a specialized language called_____	Grammar	English-like language	Object Oriented Language	Structured Query Language	**Structured Query Language**
24	_____ are the one who really use and take the benefits of database.	Database administrators	Database users	Database assistants	Database associates	**Database users**
25	DCL stands for _____	Data Consistency Language	Data Commit Language	Data Concurrency Language	Data Control Language	**Data Control Language**
26	_____is any one execution of a user program in a DBMS	Compiled program	transaction	user output	concurrent execution	**transaction**
27	_____is a mechanism used to control access to database objects	stop	terminate	Authorization	Restriction	**Authorization**
28	The task come under DML is _____	Select	Commit	Grant	Create	**Select**
29	The term DBA is _____	Database Assistant	Database Administrator	Departmental Based Assistant	Data Based Administration	**Database Administrator**
30	_____is a collection of pages or a collection of records.	Indexes	record blocks	files	Query	**files**
31	In DBMS, which one contains metadata i.e., data about the database?	Database	RDBMS	Data Dictionary	DBA	**Data Dictionary**
32	_____is an object in the real world	Entity	Attribute	Relationship	Property	**Entity**
33	Collection of similar entities are called_____	Attributes	Entity	Entity Set	Relationship	**Entity Set**
34	An Entity is described using a set of _____	Entity	Entity Set	Attributes	Relationship	**Attributes**

35	_____is used to uniquely identify an entity in the set.	Key	Lock	Attributes	Entity	**Key**
36	_____is used to uniquely identify a particular row	Candidate Key	Primary Key	foreign Key	Referential key	**Primary Key**
37	A _____is an association among two or more entities	Attributes	Entity Sets	Key	Relationships	**Relationships**
38	Indicated by using arrow from entities to relationships in the ER diagram.	Arrow	Thick line	Dotted line	Shaded line	**Arrow**
39	Derived attribute is indicated by _____in ER diagram	Solid line	Thick line	Thin line	Dotted line	**Dotted line**
40	_____is a set of associated values	Entity	Attribute	Relationships	Domain	**Domain**
41	_____consists of a relation schema and a relation instance.	relation	table	domain	entity	**relation**
42	An instance of a relation is a set of _____	tuple	domain	attribute	relationships	**tuple**
43	Each tuple is a _____	Column	row	table	instance	**row**
44	Which among the following is not used to develop Application interface?	Power Builder	Visual Basic	Oracle	Developer 2000	**Oracle**
45	Centralized control of the database is exerted by _____ user	Naïve	Online	DBA	Application Programmer	**DBA**
46	_____ users communicate with the database directly via an online terminal	Naïve	Online	DBA	Application Programmer	**Online**

47	_____ is not the ACID property.	Atomicity	Consistency	Isolation	Desirability	**Desirability**
48	Data Dictionary is also called _____	Data files	Meta Data	Data compiler	Data manager	**Meta Data**
49	The data dictionary is automatically updated by the database management system when any changes are made in the database is known as	Passive Data Dictionary	Active Data Dictionary	Intellegent Data Dictionay	Independent Data Dictionay	**Active Data Dictionary**
50	Which one is the task of TCL?	Grant	Create	Drop	Rollback	**Rollback**
51	_____ is the process of reconstructing the data in case of failure	Recovery	Backup	Integrity	Consistency	**Recovery**
52	_____ is a logically interrelated collection of shared data physically distributed over a computer network.	Database	Relational Database	Object oriented database	Distributed Database	**Distributed Database**
53	_____ relationship uses the keyword 'must'	Optional	Mandatory	Recursive	Many-many	**Mandatory**
54	_____ relationship uses the keyword 'may'	Optional	Mandatory	Recursive	Many-many	**Optional**
55	_____ is a circular relationship that exists between two attributes in the same entity.	Optional	Mandatory	Recursive	Many-many	**Recursive**
56	The degree of a relation is also called _____	arity	instance	relation	Schema	**arity**
57	A _____ is the person who accepts the privileges provided to him	Granter	Grantee	Administrator	Revoker	**Grantee**
58	_____ keyword is used to provide a specific privilege to all the users of database.	Public	All	All Users	Public users	**Public**

59	____ the command used to withdraw the access privilege given already	Withdraw	Revoke	Remove	Extract	**Revoke**
60	____ the command used to provide access privilege to the user.	Withdraw	Revoke	Grant	Allow	**Grant**

UNIT-II

SYLLABUS

Data Models, types and their comparison, Entity Relationship Model, Entity Types, Entity Sets, Attributes and its types, Keys, E-R Diagram, Data Integrity RDBMS –Concept, Components and Codd's rules.

.

DBMS Database Models

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

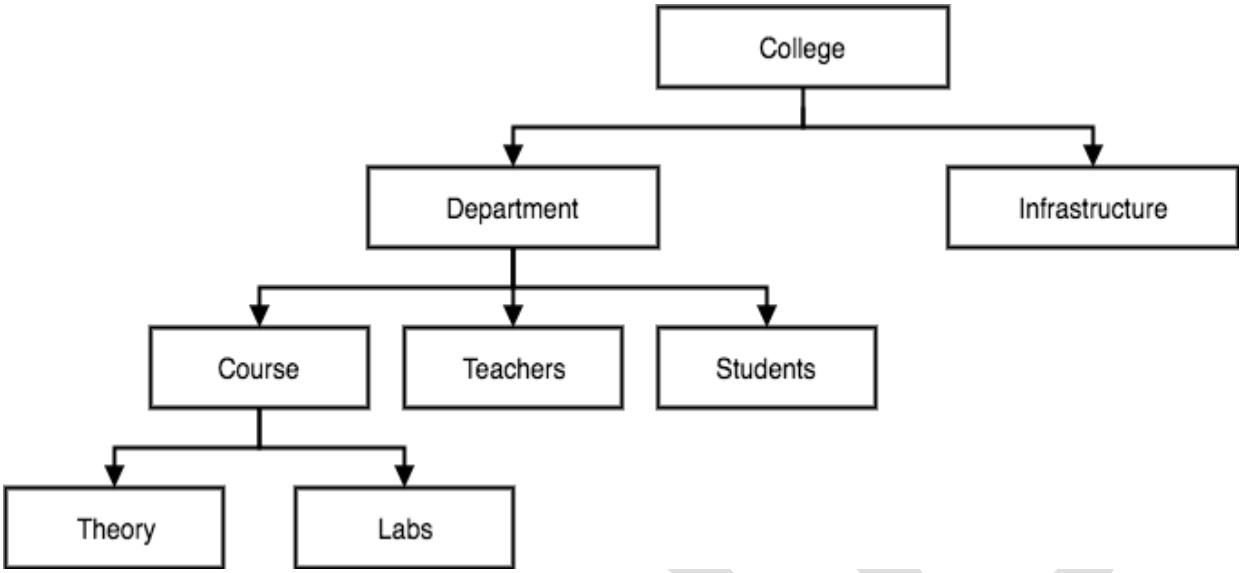
- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

Hierarchical Model

This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The heirarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node. This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organised into tree-like structure with one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and many students.

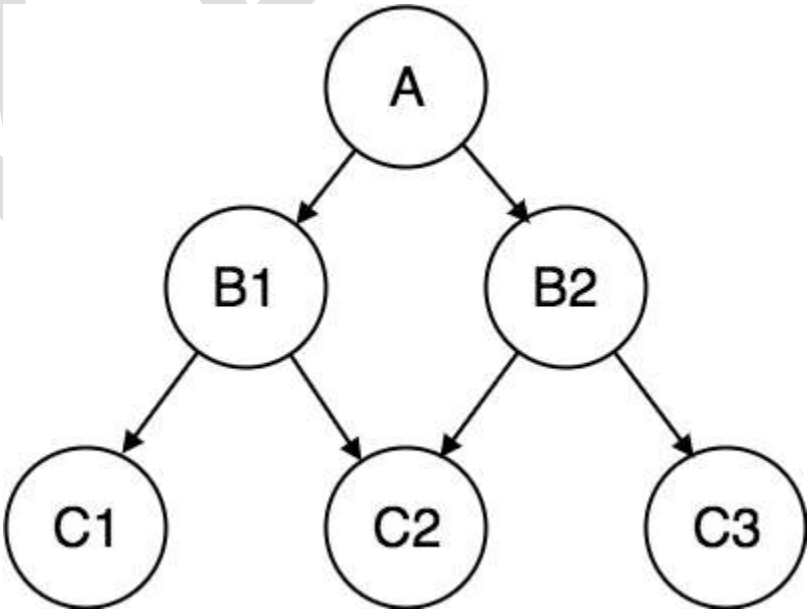


Network Model

This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.



Entity-relationship Model

In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

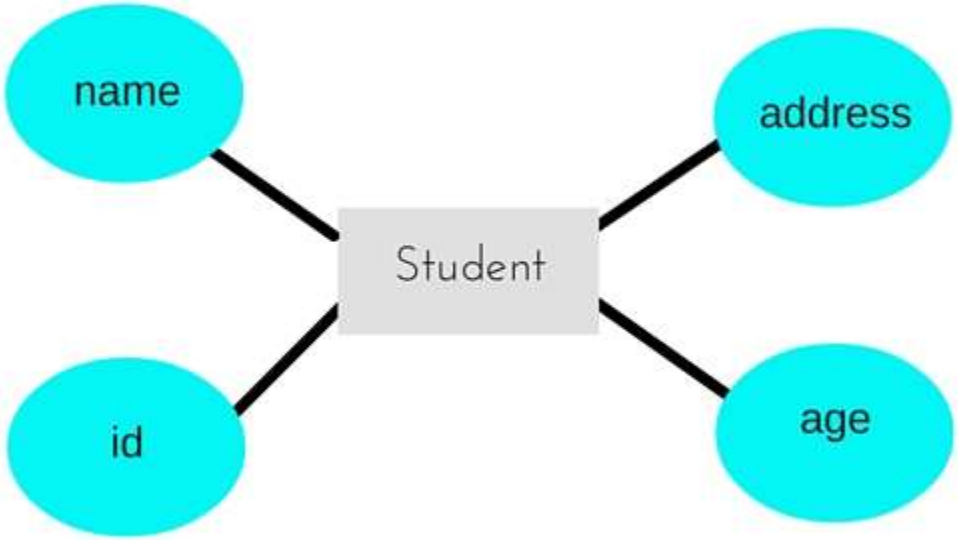
Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

This model is good to design a database, which can then be turned into tables in relational model.

Let's take an example, if we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.

Relationships can also be of different types.



Relational Model

In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, in fact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table. Hence, tables are also known as **relations** in relational model.

student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

Entity Relationship Model

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

Weak Entity

A weak Entity is represented using double rectangular boxes. It is generally connected to another entity.

Entity Sets

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

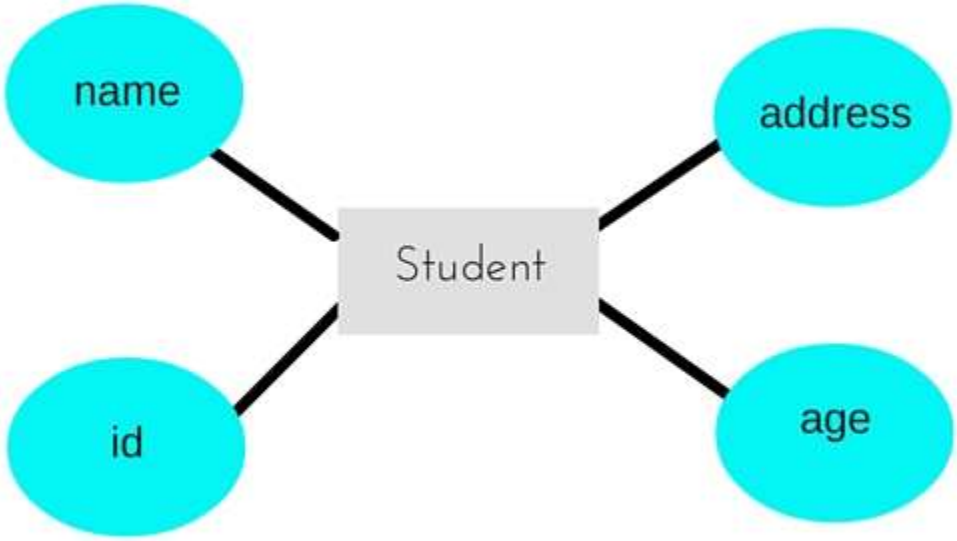
For example, the roll_number of a student makes him/her identifiable among students.

- **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Attributes

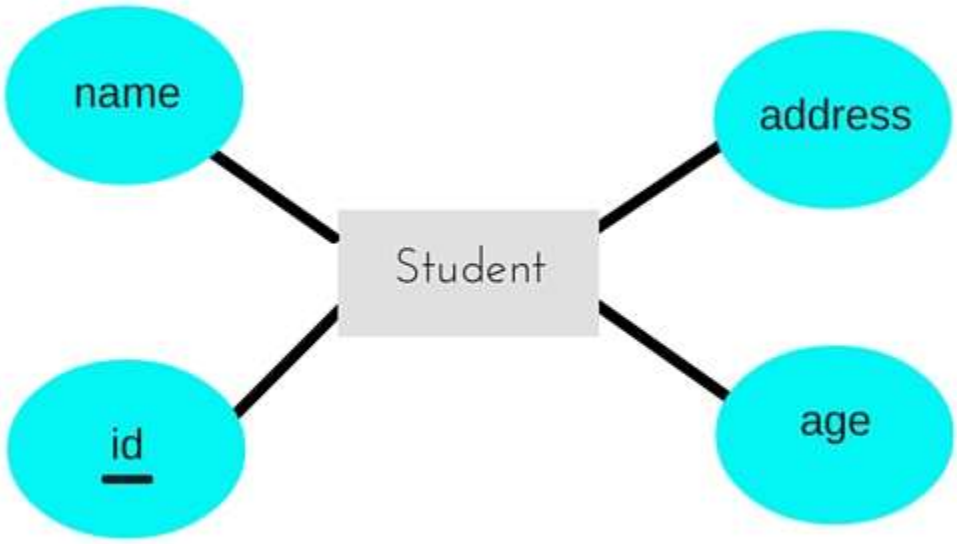
An Attribute describes a property or characteristic of an entity. For example, Name, Age, Address etc can be attributes of a Student. An attribute is represented using eclipse.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.



Key Attribute

Key attribute represents the main characteristic of an Entity. It is used to represent a Primary key. To represent a Key attribute, the attribute name inside the Ellipse is underlined. For example − Social_Security_Number.



Simple attribute

Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.

Derived Attribute

Derived attributes are those which are derived based on other attributes. Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.

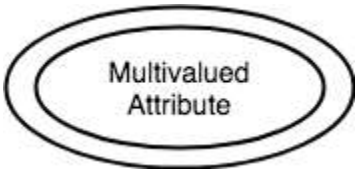
To represent a derived attribute, another dotted ellipse is created inside the main ellipse.

Single-value attribute

Single-value attributes contain single value. For example − Social_Security_Number.

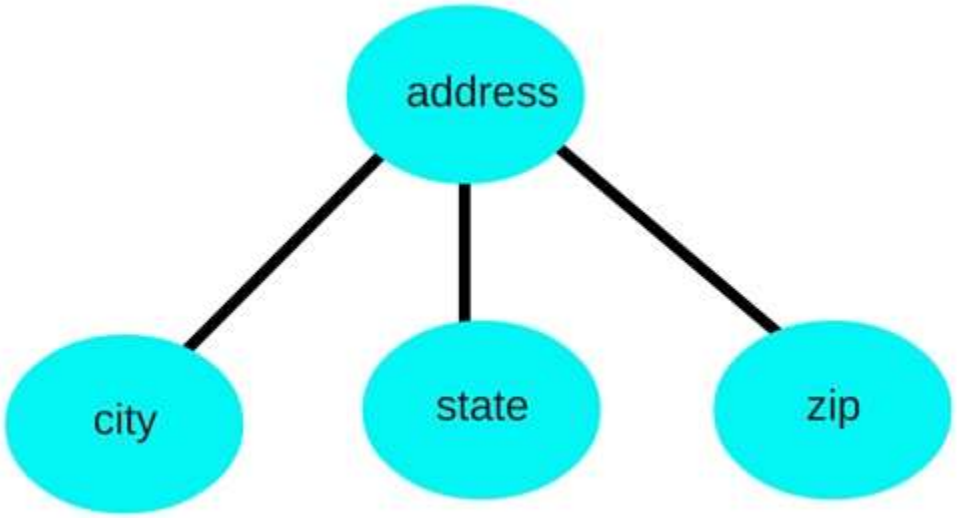
Multivalued Attribute

Double Ellipse, one inside another, represents the attribute which can have multiple values. Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.



Composite Attribute

A composite attribute is the attribute, which can also have their own attributes. Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.



These attribute types can come together in a way like −

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

Relationship

The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

Relationship Set

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

There are three types of relationship that exist between Entities.

1. Binary Relationship
2. Recursive Relationship
3. Ternary Relationship

Binary Relationship

Binary Relationship means relation between two Entities. This is further divided into three types.

One to One Relationship

This type of relationship is rarely seen in real world.

The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in real-world relationships.

One to Many Relationship

The below example showcases this relationship, which means that 1 student can opt for many courses, but a course can only have 1 student. Sounds weird! This is how it is.

Many to One Relationship

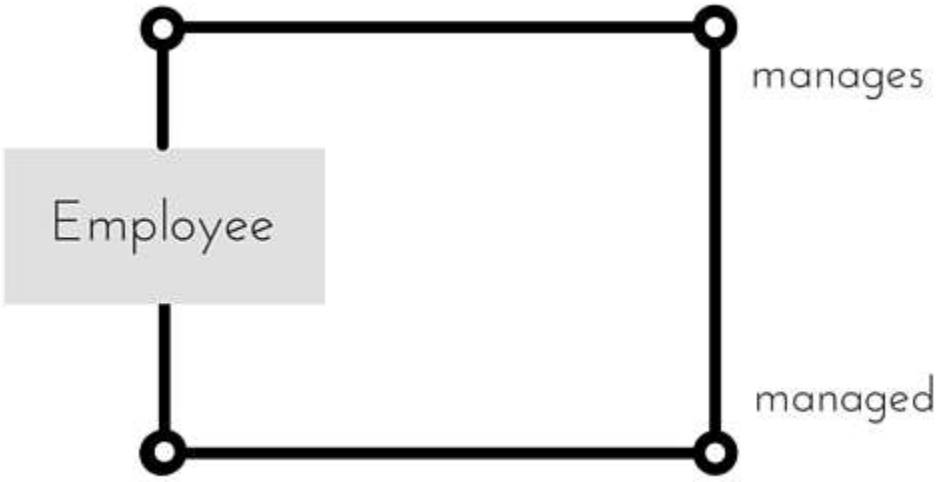
It reflects business rule that many entities can be associated with just one entity. For example, Student enrolls for only one Course but a Course can have many Students.

Many to Many Relationship

The above diagram represents that one student can enroll for more than one courses. And a course can have more than 1 student enrolled in it.

Recursive Relationship

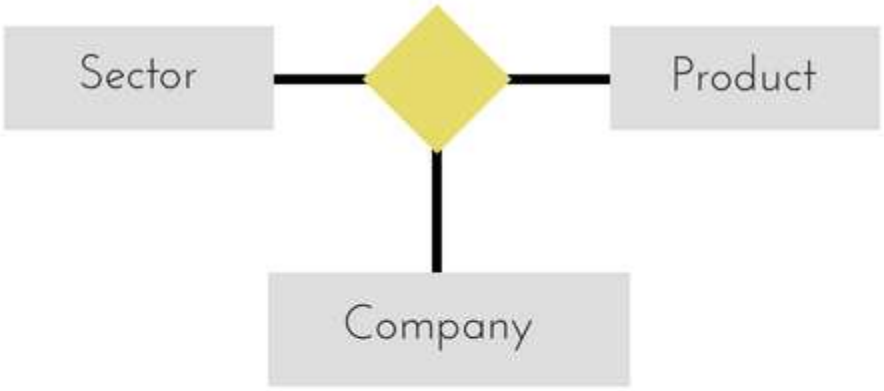
When an Entity is related with itself it is known as **Recursive** Relationship.



Ternary Relationship

Relationship of degree three is called Ternary relationship.

A Ternary relationship involves three entities. In such relationships we always consider two entites together and then look upon the third.



- The above relationship involves 3 entities.
- Company operates in Sector, producing some Products.

For example, in the diagram above, we have three related entities, **Company**, **Product** and **Sector**. To understand the relationship better or to define rules around the model, we should relate two entities and then derive the third one.

A **Company** produces many **Products**/ each product is produced by exactly one company.

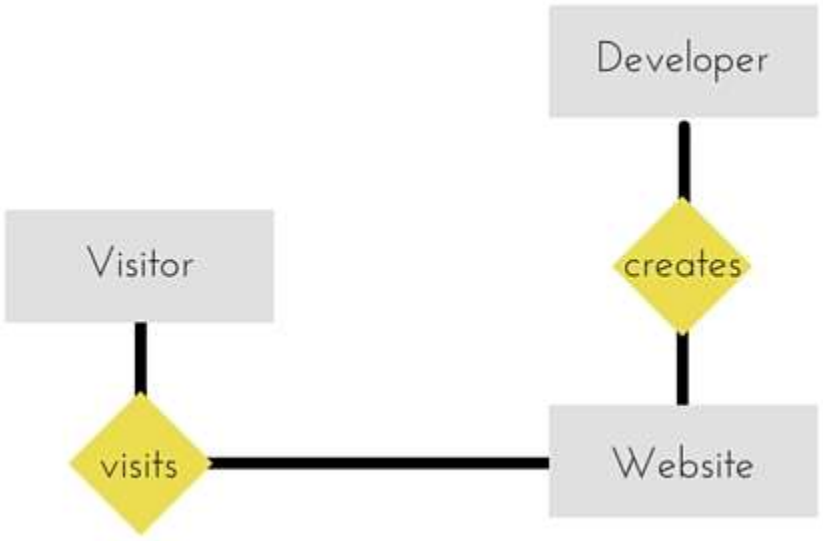
A **Company** operates in only one **Sector** / each sector has many companies operating in it.

Considering the above two rules or relationships, we see that although the complete relationship involves three entities, but we are looking at two entities at a time.

E-R Diagram

ER Diagram is a visual representation of data that describes how data is related to each other. In ER Model, we disintegrate data into entities, attributes and setup relationships between entities, all this can be represented visually using the ER diagram.

For example, in the below diagram, anyone can see and understand what the diagram wants to convey: *Developer develops a website, whereas a Visitor visits a website*.



Entitiy, Attributes, Relationships etc form the components of ER Diagram and there are defined symbols and shapes to represent each one of them.

Let's see how we can represent these in our ER Diagram.

Entity

An **Entity** can be any object, place, person or class. In ER Diagram, an **entity** is represented using rectangles. Consider an example of an Organisation- Employee, Manager, Department, Product and many more can be taken as entities in an Organisation.

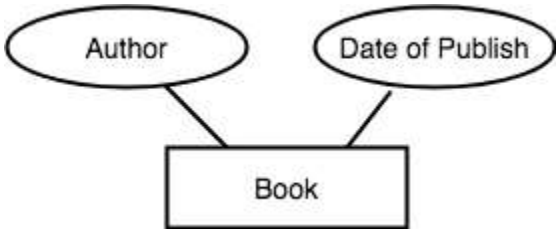
The yellow rhombus in between represents a relationship.

Relationship

A Relationship describes relation between **entities**. Relationship is represented using diamonds or rhombus.

Attributes for any Entity

Ellipse is used to represent attributes of any entity. It is connected to the entity.



Data Integrity

Data integrity is the overall completeness, accuracy and consistency of data. This can be indicated by the absence of alteration between two instances or between two updates of a data record, meaning data is intact and unchanged. Data integrity is usually imposed during the database design phase through the use of standard procedures and rules. Data integrity can be maintained through the use of various error-checking methods and validation procedures. Data integrity is enforced in both hierarchical and relational database models. The following three integrity constraints are used in a relational database structure to achieve data integrity:

- Entity Integrity: This is concerned with the concept of primary keys. The rule states that every table must have its own primary key and that each has to be unique and not null. No two rows can be the same.
- Referential Integrity: This is the concept of foreign keys. When two or more tables have a relationship, we have to ensure that the foreign key value matches the primary key value at all times. The rule states that the foreign key value can be in two states. The first state is that the foreign key value would refer to a primary key value of another table, or it can be null. Being null could simply mean that there are no relationships, or that the relationship is unknown.

  So referential integrity will prevent users from:

  - ✓ Adding records to a related table if there is no associated record in the primary table.
  - ✓ Changing values in a primary table that result in orphaned records in a related table.

- ✓ Deleting records from a primary table if there are matching related records.

- Domain Integrity: This states that all columns in a relational database are in a defined domain. *Domain integrity* concerns the validity of entries for a given column. Selecting the appropriate data type for a column is the first step in maintaining domain integrity. Other steps could include, setting up appropriate constraints and rules to define the data format and/or restricting the range of possible values.
- User-Defined Integrity: User-defined integrity allows the user to apply business rules to the database that aren't covered by any of the other three data integrity types.

The concept of data integrity ensures that all data in a database can be traced and connected to other data. This ensures that everything is recoverable and searchable. Having a single, well-defined and well-controlled data integrity system increases stability, performance, reusability and maintainability. If one of these features cannot be implemented in the database, it must be implemented through the software.

RDBMS Concepts

RDBMS stands for relational database management system. A relational model can be represented as a table of rows and columns. A relational database has following major components:

1. Table
2. Record or Tuple
3. Field or Column name or Attribute
4. Domain
5. Instance
6. Schema
7. Keys

1. Table

A table is a collection of data represented in rows and columns. Each table has a name in database. For example, the following table "STUDENT" stores the information of students in database.

Table: STUDENT

Student_Id	Student_Name	Student_Addr	Student_Age
101	Chaitanya	Dayal Bagh, Agra	27
102	Ajeet	Delhi	26
103	Rahul	Gurgaon	24
104	Shubham	Chennai	25

2. Record or Tuple

Each row of a table is known as record. It is also known as tuple. For example, the following row is a record that we have taken from the above table.

102	Ajeet	Delhi	26

3. Field or Column name or Attribute

The above table "STUDENT" has four fields (or attributes): Student_Id, Student_Name, Student_Addr & Student_Age.

4. Domain

A domain is a set of permitted values for an attribute in table. For example, a domain of month-of-year can accept January, February,…December as values, a domain of dates can accept all possible valid dates etc. We specify domain of attribute while creating a table.

An attribute cannot accept values that are outside of their domains. For example, in the above table "STUDENT", the Student_Id field has integer domain so that field cannot accept values that are not integers for example, Student_Id cannot has values like, "First", 10.11 etc.

5. Instance and Schema

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

Definition of 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.

6. Keys

Key plays an important role in relational database; it is used for identifying unique rows from table. It also establishes relationship among tables.

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 combination of columns that uniquely identifies any row within a table.

Candidate Key – A candidate key is a closely related concept where the super key is reduced to the minimum number of columns required to uniquely identify each row.

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.

Components of DBMS

DBMS have several components, each performing very significant tasks in the database management system environment. Below is a list of components within the database and its environment.

Software

This is the set of programs used to control and manage the overall database. This includes the DBMS software itself, the Operating System, the network software being used to share the data among users, and the application programs used to access data in the DBMS.

Hardware

Consists of a set of physical electronic devices such as computers, I/O devices, storage devices, etc., this provides the interface between computers and the real world systems.

Data

DBMS exists to collect, store, process and access data, the most important component. The database contains both the actual or operational data and the metadata.

Procedures

These are the instructions and rules that assist on how to use the DBMS, and in designing and running the database, using documented procedures, to guide the users that operate and manage it.

Database Access Language

This is used to access the data to and from the database, to enter new data, update existing data, or retrieve required data from databases. The user writes a set of appropriate commands in a database access language, submits these to the DBMS, which then processes the data and generates and displays a set of results into a user readable form.

Query Processor

This transforms the user queries into a series of low level instructions. This reads the online user's query and translates it into an efficient series of operations in a form capable of being sent to the run time data manager for execution.

Run Time Database Manager

Sometimes referred to as the database control system, this is the central software component of the DBMS that interfaces with user-submitted application programs and queries, and handles database access at run time. Its function is to convert operations in user's queries. It provides control to maintain the consistency, integrity and security of the data.

Data Manager

Also called the cache manger, this is responsible for handling of data in the database, providing a recovery to the system that allows it to recover the data after a failure.

Database Engine

The core service for storing, processing, and securing data, this provides controlled access and rapid transaction processing to address the requirements of the most demanding data consuming applications. It is often used to create relational databases for online transaction processing or online analytical processing data.

Data Dictionary

This is a reserved space within a database used to store information about the database itself. A data dictionary is a set of read-only table and views, containing the different information about the data used in the enterprise to ensure that database representation of the data follow one standard as defined in the dictionary.

Report Writer

Also referred to as the report generator, it is a program that extracts information from one or more files and presents the information in a specified format. Most report writers allow the user to select records that meet certain conditions and to display selected fields in rows and columns, or also format the data into different charts.

Codd's Rules

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence

The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore – 641 021
(For the candidates admitted in 2017 onwards)

	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____is a collection of high-level data description constructs that hide many low-level storage details	Data Model	ER Model	Network Model	none	**Data Model**
2	Database management System that exhibits related tables are based on _____	Network model	Hierarchical model	Relational kmodel	Object-based model	**Relational model**
3	_____ key is used to implement Entity integrity	Primary key	Foreign key	Candidate key	Composite key	**Primary key**
4	Which key is used to implement Referential integrity	Primary key	Foreign key	Candidate key	Composite key	**Foreign key**
5	A model that pictorially represents a schema of a table is called _____	Network model	ER Model	Object-based model	Hierarchical model	**ER Model**
6	In E-R Model entities are represented by _____	Rectangle	Rhombus	Ellipse	Square	**Rectangle**
7	The database model organises data into a tree-like-structure, with a single root, to which all the other data is linked is	Network Model	Relational Model	E-R Model	Hierarchical Model	**Hierarchical Model**
8	In _____ relationship one record in a table can be related to many records in another table	one-many	one-one	many-many	many-one	**one-many**
9	A minimal set of attributes that determines the entire tuple is _____	Super key	Foreign key	Candidate key	Primary key	**Candidate key**
10	Graphical representation of an entity is called	Data Flow Diagram	Structure Diagram	Use case Diagram	ER Diagram	**ER Diagram**

11	_____ is a single instance in an entity collection	Entity	Entity set	Entity instance	Entity collection	**Entity instance**
12	____ attributes cannot be further divided into smaller components	Composite	Simple	Derived	Stored	**Simple**
13	Which attribute have more than one value for an entity instance	Derived	Simple	Composite	Multi-valued	**Multi-valued**
14	Derived attributes are depicted in the E-R diagram with	double-lined ellipse	single-lined ellipse	dotted line	dashed ellipse	**dashed ellipse**
15	___ is a single attribute or a combination of attributes that uniquely identify an individual instance of an entity type	Composite	Simple	Key	Derived	**Key**
16	What symbol is used to represent multi-valued attribute	double-lined ellipse	single-lined ellipse	dotted line	thick line	**double-lined ellipse**
17	_____is an object in the real world	Entity	Attribute	Relationship	Property	**Entity**
18	Collection of similar entities are called_____	Attributes	Entity	Entity Set	Relationship	**Entity Set**
19	The set of allowable values for the attribute is called	Entity	Domain	Table	Degree	**Domain**
20	An Entity is described using a set of _____	Entity	Entity Set	Attributes	Relationship	**Attributes**
21	_____is used to uniquely identify an entity in the set.	Key	Lock	Attributes	Entity	**Key**
22	_____is used to uniquely identify a particular row	Candidate Key	Primary Key	foreign Key	Referential key	**Primary Key**
23	A relation with degree n is called as	Unary	Binary	Ternary	n - ary	**n - ary**
24	A _____is an association among two or more entities	Attributes	Entity Sets	Key	Relationships	**Relationships**

25	Indicated by using arrow from entities to relationships in the ER diagram.	Arrow	Thick line	Dotted line	Shaded line	**Arrow**
26	Derived attribute is indicated by _____in ER diagram	Solid line	Thick line	Thin line	Dotted line	**Dotted line**
27	_____is a set of associated values	Entity	Attribute	Relationships	Domain	**Domain**
28	_____consists of a relation schema and a relation instance.	relation	table	domain	entity	**relation**
29	An instance of a relation is a set of _____	tuple	domain	attribute	relationships	**tuple**
30	Each tuple is a _____	Column	row	table	instance	**row**
31	_____ is the most widely used data model.	Network Model	E-R Model	Hierarchical Model	Relational Model	**Relational Model**
32	_____ is a visual representation of data that describes how data is related to each other.	Entity Sets	Key Attribute	Binary Relationship	E-R Diagram	**E-R Diagram**
33	The tables are also called as _____	Relations	Entities	E-R Model	Network Model	**Relations**
34	If only one entity is involved in a relation it is called _____	Unary	Binary	Ternary	None	**Unary**
35	If three entities are involved in a relation it is called _____	Unary	Binary	Ternary	None	**Ternary**
36	If two entities are involved in a relation it is called _____	Unary	Binary	Ternary	None	**Binary**
37	_____is the count of the number of entities involved in that relationship	degree	cardinality	entity set	entity type	**cardinality**
38	The degree of a relation is also called _____	arity	instance	relation	Schema	**arity**
39	_____ is a circular relationship that exists between two attributes in the same entity.	Optional	Mandatory	Recursive	Many-many	**Recursive**

40	Selection Operation is used to _____from a relation	Select the columns	Select the rows	Select the table	none.	**Select the rows**
41	A relation is represented conceptually as a	Matrix	File	Table	Record	**Table**
42	Which one of the following can be a primary key for Employee Database	Address	Age	Date of Birth	Employee ID	**Employee ID**
43	The name for a relationship must always be	a noun	a verb	an adjective	a preposition	**a verb**
44	The name for an entity must always be	a noun	a verb	an adjective	a preposition	**a noun**
45	A weak Entity is represented using _____ and is generally connected to another entity.	Circle	Ellipse	double rectangular boxes	Rectangle	**double rectangular boxes**
46	A set of attributes (one or more) that collectively identifies an entity in an entity set is called as _____	Primary Key	Candidate Key	Foreign Key	Super Key	**Super Key**
47	Selection Operation is used to _____from a relation	Select the columns	Select the rows	Select the table	none.	**Select the rows**
48	_____ is the attribute, which can also have their own attributes.	Key Attribute	Derived Attribute	Composite Attribute	Simple Attribute	**Composite Attribute**
49	In _____ Model, as the data is more related, hence accessing the data is also easier and fast and also the database model was used to map many-to-many data relationships.	Nerwork	E - R	Hierarchical	Relational	**Nerwork**
50	The _____ model was introduced by E.F Codd in 1970.	Nerwork	E - R	Hierarchical	Relational	**Relational**
51	_____ represents the main characteristic of an Entity.	Key Attribute	Derived Attribute	Composite Attribute	Simple Attribute	**Key Attribute**
52	_____statement is used to define a new table.	Create	Produce	Insert	Add	**Create**

53	Tuples are inserted using the _____command	Create	Insert	Add	Make	**Insert**
54	Tuples are deleted using the _____command	Delete	drop	remove	alter	**Delete**
55	Modify the column values in an existing row using_____ command	Modify	Alter	Update	Change	**Update**
56	_____clause is used to modify a particular row.	Update	Modify	Alter	Where	**Update**
57	Every relation must have a	Primary key	Foreign key	Reference key	Composite key	**Primary key**
58	The operations and rules for a relation is defined by	Kevin	E.F.Codd	Gerald	George	**E.F.Codd**
59	_____is a condition specified on a database schema & restricts the data that can be stored in an instance of the database.	integrity Constraint	restriction	key	none	**integrity Constraint**
60	A set of fields that uniquely identifies a tuple according to a key constraint is called_____ for the relation	primary key	Candidate key	foreign key	Super key	**Candidate key**

UNIT-III
SYLLABUS

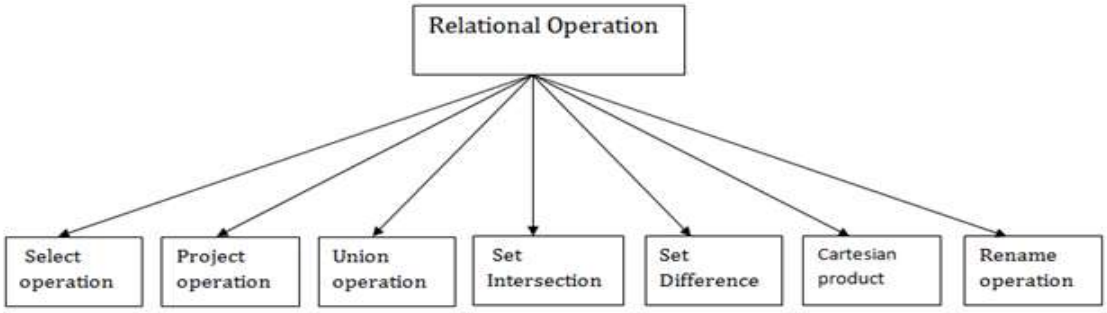
Relational Algebra (selection, projection, union, intersection, Cartesian product, Different types of join like theta join, equi-join, natural join, outer join) Functional Dependencies, Good & Bad Decomposition, Anomalies as a database: A consequences of bad design, Normalization: 1NF, 2NF, 3NF, BCNF, 4NF 5NF.

Relational database systems are expected to be equipped with a query language that can assist its users to query the database instances. There are two kinds of query languages − relational algebra and relational calculus.

Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

Types of Relational operation



The fundamental operations of relational algebra are as follows −

- Select
- Project

- Union
- Set different
- Cartesian product
- Rename

We will discuss all these operations in the following sections.

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation − $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.

For example −

$\sigma_{subject = "database"}(Books)$

Output − Selects tuples from books where subject is 'database'.

$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$

Output − Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$

Output − Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

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:

σ BRANCH_NAME="perryride" (LOAN)

Output:

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

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation − $\prod_{A1, A2, An}$ (r)

Where A_1, A_2 , A_n are attribute names of relation **r**.

Duplicate rows are automatically eliminated, as relation is a set.

For example −

$\prod_{subject, author}$ (Books)

Selects and projects columns named as subject and author from the relation Books.

CUSTOMER RELATION

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

Input:

∏ NAME, CITY (CUSTOMER)

Output:

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

Union Operation (∪)

It performs binary union between two given relations and is defined as −

$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$

Notation − r U s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$\prod_{author}$ (Books) ∪ $\prod_{author}$ (Articles)

Output − Projects the names of the authors who have either written a book or an article or both.

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:

∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson

Curry
Williams
Mayes

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation − **r − s**

Finds all the tuples that are present in **r** but not in **s**.

$\prod_{author}$ (Books) − $\prod_{author}$ (Articles)

Output − Provides the name of authors who have written books but not articles.

Using the above DEPOSITOR table and BORROW table

Input:

∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

Cartesian Product (X)

Combines information of two different relations into one.

Notation − r X s

Where **r** and **s** are relations and their output will be defined as −

r X s = { q t | q ∈ r and t ∈ s}

$\sigma_{author = 'balamurugan'}$(Books X Articles)

Output − Yields a relation, which shows all the books and articles written by balamurugan.

Cross JOIN Syntax is,

SELECT column-name-list
FROM
table-name1 CROSS JOIN table-name2;

Example of Cross JOIN

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:

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

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** *ρ*.

Notation − ρ_x (E)

Where the result of expression **E** is saved with name of **x**.

Additional operations are −

- Set intersection
- Assignment
- Natural join

SQL Joins

SQL Joins are used to relate information in different tables. A Join condition is a part of the sql query that retrieves rows from two or more tables. A SQL Join condition is used in the SQL WHERE Clause of select, update, delete statements.

The Syntax for joining two tables is:

```
SELECT col1, col2, col3...FROM table_name1, table_name2
WHERE table_name1.col2 = table_name2.col1;
```

If a sql join condition is omitted or if it is invalid the join operation will result in a Cartesian product. The Cartesian product returns a number of rows equal to the product of all rows in all the tables being joined. For example, if the first table has 20 rows and the second table has 10 rows, the result will be 20 * 10, or 200 rows. This query takes a long time to execute.

Lets use the below two tables to explain the sql join conditions.

database table "product";

product_id	product_name	supplier_name	unit_price
100	Camera	Nikon	300
101	Television	Onida	100
102	Refrigerator	Videocon	150
103	Ipod	Apple	75
104	Mobile	Nokia	50

database table "order_items";

order_id	product_id	total_units	customer
5100	104	30	Infosys
5101	102	5	Satyam
5102	103	25	Wipro
5103	101	10	TCS

SQL Joins can be classified into Equi join and Non Equi join.

1) SQL Equi joins

It is a simple sql join condition which uses the equal sign as the comparison operator. Two types of equi joins are SQL Outer join and SQL Inner join. For example: We can get the information about a customer who purchased a product and the quantity of product.

2) SQL Non equi joins

It is a sql join condition which makes use of some comparison operator other than the equal sign like >, <, >=, <=

1) SQL Equi Joins:

An equi-join is further classified into two categories:

i) SQL Inner Join

ii) SQL Outer Join

a) SQL INNER Join or EQUI Join:

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Inner Join Syntax is,

SELECT column-name-list FROM
table-name1 INNER JOIN table-name2
WHERE table-name1.column-name = table-name2.column-name;

All the rows returned by the sql query satisfy the sql join condition specified.

For example: If we want to display the product information for each order the query will be as given below. Since you are retrieving the data from two tables, you need to identify the common column between these two tables, which is the product_id.

The query for this type of sql joins would be like,

SELECT order_id, product_name, unit_price, supplier_name, total_units FROM product, order_items WHERE order_items.product_id = product.product_id;

The columns must be referenced by the table name in the join condition, because product_id is a column in both the tables and needs a way to be identified. This avoids ambiguity in using the columns in the SQL SELECT statement.

The number of join conditions is (n-1), if there are more than two tables joined in a query where 'n' is the number of tables involved. The rule must be true to avoid Cartesian product.

We can also use aliases to reference the column name, then the above query would be like,

SELECT o.order_id, p.product_name, p.unit_price, p.supplier_name, o.total_units FROM product p, order_items o WHERE o.product_id = p.product_id;

Example of INNER JOIN

Consider a **class** table,

ID NAME

1 abhi

2 adam

3 alex
4 anu

and the **class_info** table,

ID Address
1 DELHI
2 MUMBAI
3 CHENNAI

Inner JOIN query will be,

SELECT * from class INNER JOIN class_info where class.id = class_info.id;

The resultset table will look like,

ID NAME ID Address
1 abhi 1 DELHI
2 adam 2 MUMBAI
3 alex 3 CHENNAI

b) SQL Self Join:

A Self Join is a type of sql join which is used to join a table to itself, particularly when the table has a FOREIGN KEY that references its own PRIMARY KEY. It is necessary to ensure that the join statement defines an alias for both copies of the table to avoid column ambiguity.
The below query is an example of a self join,
SELECT a.sales_person_id, a.name, a.manager_id, b.sales_person_id, b.name FROM sales_person a, sales_person b WHERE a.manager_id = b.sales_person_id;

2) SQL Non Equi Join:

A Non Equi Join is a SQL Join whose condition is established using all comparison operators except the equal (=) operator. Like >=, <=, <, >
For example: If you want to find the names of students who are not studying either Economics, the sql query would be like,
SELECT first_name, last_name, subject FROM student_details WHERE subject != 'Economics'
The output would be something like,

first_name	last_name	subject
Anjali	Bhagwat	Maths
Shekar	Gowda	Maths
Rahul	Sharma	Science
Stephen	Fleming	Science

We understand the benefits of taking a Cartesian product of two relations, which gives us all the possible tuples that are paired together. But it might not be feasible for us in certain cases to take a Cartesian product where we encounter huge relations with thousands of tuples having a considerable large number of attributes.

Join is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

We will briefly describe various join types in the following sections.

Theta (θ) Join

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol **θ**.

Notation

$R1 \bowtie_\theta R2$

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don't have anything in common, that is $R1 \cap R2 = \Phi$.

Theta join can use all kinds of comparison operators.

Student

SID	Name	Std
101	Alex	10
102	Maria	11

Subjects

Class Subject

10 Math

10 English

11 Music

11 Sports

Student_Detail −

STUDENT $\bowtie_{Student.Std = Subject.Class}$ SUBJECT

Student_detail

SID Name Std Class Subject

101 Alex 10 10 Math

101 Alex 10 10 English

102 Maria 11 11 Music

102 Maria 11 11 Sports

Natural Join (⋈)

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

The syntax for Natural Join is,

SELECT * FROM
table-name1 NATURAL JOIN table-name2;

Example of Natural JOIN

Here is the **class** table,

ID NAME

1 abhi

2 adam

3 alex
4 anu

and the **class_info** table,

ID Address
1 DELHI
2 MUMBAI
3 CHENNAI

Natural join query will be,

SELECT * from class NATURAL JOIN class_info;

The resultset table will look like,

ID NAME Address
1 abhi DELHI
2 adam MUMBAI
3 alex CHENNAI

In the above example, both the tables being joined have **ID** column(same name and same datatype), hence the records for which value of **ID** matches in both the tables will be the result of Natural Join of these two tables.

OUTER JOIN

This sql join condition returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables. The sql outer join operator in Oracle is (+) and is used on one side of the join condition only.
The syntax differs for different RDBMS implementation. Few of them represent the join conditions as "sql left outer join", "sql right outer join".
If you want to display all the product data along with order items data, with null values displayed for order items if a product has no order item, the sql query for outer join would be as shown below:
SELECT p.product_id, p.product_name, o.order_id, o.total_units FROM order_items o, product p WHERE o.product_id (+) = p.product_id;

The output would be like,

product_id	product_name	order_id	total_units
100	Camera		
101	Television	5103	10
102	Refrigerator	5101	5
103	Ipod	5102	25
104	Mobile	5100	30

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

1. Left Outer Join
2. Right Outer Join
3. Full Outer Join

LEFT Outer Join

The left outer join returns a resultset table with the **matched data** from the two tables and then the remaining rows of the **left** table and null from the **right** table's columns.

Syntax for Left Outer Join is,

```
SELECT column-name-list FROM
table-name1 LEFT OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;
```

To specify a condition, we use the ON keyword with Outer Join.

Left outer Join Syntax for **Oracle** is,

```
SELECT column-name-list FROM
table-name1, table-name2 on table-name1.column-name = table-name2.column-name(+);
```

Example of Left Outer Join

Here is the **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Left Outer Join query will be,

```
SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id = class_info.id);
```

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null

RIGHT Outer Join

The right outer join returns a resultset table with the **matched data** from the two tables being joined, then the remaining rows of the **right** table and null for the remaining **left** table's columns.

Syntax for Right Outer Join is,

SELECT column-name-list FROM
table-name1 RIGHT OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;

Right outer Join Syntax for **Oracle** is,

SELECT column-name-list FROM
table-name1, table-name2
ON table-name1.column-name(+) = table-name2.column-name;

Example of Right Outer Join

Once again the **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Right Outer Join query will be,

SELECT * FROM class RIGHT OUTER JOIN class_info ON (class.id = class_info.id);

The resultant table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT

Full Outer Join

The full outer join returns a resultset table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

Syntax of Full Outer Join is,

```
SELECT column-name-list FROM
table-name1 FULL OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;
```

Example of Full outer join is,

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI

2 MUMBAI
3 CHENNAI
7 NOIDA
8 PANIPAT

Full Outer Join query will be like,

SELECT * FROM class FULL OUTER JOIN class_info ON (class.id = class_info.id);

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

$$X \rightarrow Y$$

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

For example:

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

Emp_Id → Emp_Name

We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional dependency

1. Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B

Example:

1. Consider a table with two columns Employee_Id and Employee_Name.
2. {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as
3. Employee_Id is a subset of {Employee_Id, Employee_Name}.
4. Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial de pendencies too.

2. Non-trivial functional dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.

Example:

1. ID → Name,
2. Name → DOB

Decomposition

A functional **decomposition** is the process of breaking down the functions of an organization into progressively greater (finer and finer) levels of detail.
In decomposition, one function is described in greater detail by a set of other supporting functions.
The decomposition of a relation scheme R consists of replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R and together include all attributes in R.

Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistencies and anomalies.

Good & Bad Decomposition

There are two types of decomposition :

1. Lossy Decomposition
2. Lossless Join Decomposition

Lossy Decomposition :

"The decompositio of relation R into R1 and R2 is **lossy** when the join of R1 and R2 does not yield the same relation as in R."

One of the disadvantages of decomposition into two or more relational schemes (or tables) is that some information is lost during retrieval of original relation or table. Consider that we have table STUDENT with three attribute roll_no , sname and department.

STUDENT:

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation no_name and name_dept :

No_name:

Roll_no	Sname
111	parimal
222	parimal

Name_dept :

Sname	Dept

parimal	COMPUTER
parimal	ELECTRICAL

In lossy decomposition ,spurious tuples are generated when a natural join is applied to the relations in the decomposition.

stu_joined :

Roll_no	**Sname**	**Dept**
111	parimal	COMPUTER
111	parimal	ELECTRICAL
222	parimal	COMPUTER
222	parimal	ELECTRICAL

The above decomposition is a bad decomposition or Lossy decomposition.

Lossless Join Decomposition :

"The decompositio of relation R into R1 and R2 is **lossless** when the join of R1 and R2 yield the same relation as in R."
A relational table is decomposed (or factored) into two or more smaller tables, in such a way that the designer can capture the precise content of the original table by joining the decomposed parts. This is called lossless-join (or non-additive join) decomposition.
This is also refferd as non-additive decomposition.
The lossless-join decomposition is always defined with respect to a specific set F of dependencies.
Consider that we have table STUDENT with three attribute roll_no , sname and department.

STUDENT :

Roll_no	**Sname**	**Dept**

111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation Stu_name and Stu_dept :

Stu_name:

Roll_no	Sname
111	parimal
222	parimal

Stu_dept :

Roll_no	Dept
111	COMPUTER
222	ELECTRICAL

Now ,when these two relations are joined on the comman column 'roll_no' ,the resultant relation will look like stu_joined.

stu_joined :

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

In lossless decomposition, no any spurious tuples are generated when a natural joined is applied to the relations in the decomposition.

Normalization in DBMS: Anomalies, Advantages, Disadvantages: At a basic level, normalization is the simplification of any bulk quantity to an optimum value. In the digital

world, normalization usually refers to database normalization which is the process of organizing the columns (attributes) and tables (relations) of a relational database to minimize data repetition. In the process of database creation, normalization involves organizing data into optimal tables in such a way that the results obtained are always unambiguous and clear in concept.

Though database normalization can have the effect of duplication of data, it completely removes data redundancy. This process can be considered as a refinement process after the initial identification of data objects that are to be included in the database. It involves identification of the relationship between the data objects and defining the tables required and the columns to be added within each table.

This article is all about Normalization in DBMS: Anomalies, Advantages and disadvantages.

Anomalies as a database: A consequences of bad design Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

If a database design is not done properly, it may cause several anomalies to occur in it. Normalization is essential for removing various anomalies like:

Anomalies in Database

1) Update Anomalies: When several instances of the same data are scattered across the database without proper relationship/link, it could cause strange conditions where a few of the instances will get updated with new values whereas some of them will not. This leaves the database in an inconsistent state.

2) Deletion Anomalies: Incomplete deletion of a particular data section which leaves some residual instances. The database creator remains unaware of such unwanted data as it is present at a different location.

3) Insertion Anomalies: This occurs when an attempt to insert data into a non-existent record.

Paying attention to these anomalies can help to maintain a consistent database.

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.

Types of Normal Forms

Normal Form	Description

1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Super key in the table above:

1. {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007

555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283

D394	300
D283	232
D283	549

Functional dependencies:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID
For the second table: EMP_DEPT
For the third table: {EMP_ID, EMP_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing

74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math**and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket

59	Hockey

Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Example

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash

Semester 1	Praveen

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore – 641 021
(For the candidates admitted in 2017 onwards)

RDBMS Unit III One Mark Questions

No.	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____ is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key.	2NF	3NF	4NF	5NF	**4NF**
2	_____ is a procedural query language, which takes instances of relations as input and yields instances of relations as output.	Arithmatics	Relational algebra	Relational Model	Relational Calculus	**Relational algebra**
3	A table is in _____ only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.	2NF	3NF	4NF	5NF	**5NF**
4	Which of the following is relational algebra operation?	UPDATE	SELECTION	CARTESIAN PRODUCT	UNION	**UPDATE**
5	_____is a condition specified on a database schema & restricts the data that can be stored in an instance of the database.	Integrity Constraint	restriction	key	none	**Integrity Constraint**
6	If two relations have 5 and 10 tuples respectively, then what will be the number in the CARTESIAN PRODUCT?	5	10	15	50	**50**
7	_____ performs binary union between two given relations.	Project Operation	Union Operation	Rename Operation	Select Operation	**Union Operation**

8	SQL Non Equi join does not use the symbol	>	!=	=	>=	**=**
9	_____value is unknown or not applicable	not null	zero	unknown	null	**null**
10	In which normal form transitive dependency is eliminated?	1NF	2NF	3NF	4NF	**3NF**
11	_____is a specialized language for writing queries	Sub language	Object Oriented Language	Query language	Programming language	**Query language**
12	What is the expansion of BCNF?	Boyd – Crowell Normal Form	Boyce – Codd Normal Form	Boyd – Codd Normal Form	Bayes - Codd Normal Form	**Boyce – Codd Normal Form**
13	_____is the collection of schemas for the relations in the database.	Physical Schema	Conceptual Schema	Logical Schema	relational database	**relational database**
14	_____statement is used to define a new table.	Create	Produce	Insert	Add	**Create**
15	Which of the following is the process of increasing redundancy in the database either for convenience or to improve performance?	Normalization	Optimization	Demoralization	Decomposition	**Demoralization**
16	_____clause is used to modify a particular row.	Update	Modify	Alter	Where	**Update**
17	_____retain all fields of selected tuples in the result.	Star	*	X	%	*****
18	A _____ is a set of permitted values for an attribute in table.	Instance	Domain	Schema	Key	**Domain**
19	Tuples are inserted using the _____command	Create	Insert	Add	Make	**Insert**
20	Tuples are deleted using the _____command	Delete	drop	remove	alter	**Delete**
21	Modify the column values in an existing row using_____ command	Modify	Alter	Update	Change	**Update**

22	A set of fields that uniquely identifies a tuple according to a key constraint is called_____ for the relation	primary key	Candidate key	foreign key	Super key	**Candidate key**
23	_____is used to refer the primary key in another entity	Candidate key	referential key	foreign key	Primary key	**foreign key**
24	Which one of the following is not outer join?	Full Outer Join	Left Outer Join	Right Outer Join	Non Equi Join	**Non Equi Join**
25	Indexes can be created on _____	Columns	Rows	Tuples	Triggers	**Columns**
26	___ is a type of sql join which is used to join a table to itself	Inner join	Outer join	Equi join	Self join	**Self join**
27	In self join, how many tables are involved?	1	2	3	4	1
28	The + symbol is used on one side of the join condition in ____ join	Inner join	Outer join	Equi join	Non equi join	**Outer join**
29	Which one is not the component of DBMS?	Query Processor	Data Manager	Database Engine	Data User	**Data User**
30	_____ condition makes use of comparison operators other than the equal sign	Inner join	Outer join	Equi join	Non equi join	**Non equi join**
31	___ are used to relate information in different tables	Join	Selection	Projection	Intersection	**Join**
32	Which of the following aggregate function does not ignore nulls its results?	COUNT	COUNT (*)	MAX	MIN	**COUNT (*)**
33	_____ selects tuples that satisfy the given predicate from a relation.	Select Operation	Update Operation	Project Operation	Union Operation	**Select Operation**
34	Project operation results in projection of _____	Tuples	Columns	Relations	None of the above	**Columns**
35	Design of a database is called the _____.	Instance	State	View	Schema	**Schema**
36	The data stored in database at a particular moment of time is called _____ of database	Tuples	Instance	Schema	View	**Instance**
37	A table can have only one _____	Candidate key	Primary Key	Alternate key	Foreign key	**Primary Key**

38	_____ allows the user to apply business rules to the database that aren't covered by any of the other three data integrity types.	User-defined integrity	Referential Integrity	Domain Integrity	Entity Integrity	**User-defined integrity**
39	_____ concerns the validity of entries for a given column.	User-defined integrity	Referential Integrity	Entity Integrity	Domain integrity	**Domain integrity**
40	_____ is concerned with the concept of primary keys.	Entity Integrity	Referential Integrity	User-defined integrity	Domain integrity	**Entity Integrity**
41	_____clause is used to modify a particular row.	Update	Insert	Delete	Modify	**Update**
42	Any two tables that are Union-Compatible that is, have the same number of _____	Columns	rows	Columns and rows	null values	**Columns**
43	_____ is the overall completeness, accuracy and consistency of data.	Data integrity	Usefulness	Efficiency	Adaptability	**Data integrity**
44	_____ is a program that extracts information from one or more files and presents the information in a specified format.	Query Processor	Data Manager	Database Engine	Report Writer	**Report Writer**
45	Which key is used to implement referential Integrity?	Primary key	Candidate key	Foreign key	Super key	**Foreign key**
46	_____ transforms the user queries into a series of low level instructions.	Report Writer	Data Manager	Query Processor	Database Engine	**Query Processor**
47	The sql outer join operator in Oracle is _____ and is used on one side of the join condition only.	*	^	–	+	+
48	_____is a collection of high-level data description constructs that hide many low-level storage details.	Network Model	Relational Model	E-R Model	Hierarchical Model	**Relational Model**

49	_____ returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables.	Self Join	SQL Outer Join	Non Equi Join	SQL Inner Join	**SQL Outer Join**
50	_____ is a type of Inner join which is based on column having same name and same data type present in both the tables to be joined.	Natural Join	Self Join	Theta Join	Non Equi Join	**Natural Join**
51	Derived attributes are depicted in the E-R diagram with	Double-lined ellipse	Single-lined ellipse	Dotted line	Dashed ellipse	**Dashed ellipse**
52	A _____ is a type of sql join which is used to join a table to itself, particularly when the table has a FOREIGN KEY that references its own PRIMARY KEY.	Equi Join	Self Join	Non Equi Join	Theta Join	**Self Join**
53	The _____ allows us to rename the output relation.	Selection Operation	Projection Operation	Union Operation	Rename Operation	**Rename Operation**
54	The result of _____ operation is tuples, which are present in one relation but are not in the second relation.	Union Operation	Selection Operation	set difference	Projection Operation	**set difference**
55	A _____ is when changing a non-key column, might cause any of the other non-key columns to change	transitive dependency	multi-valued dependency	join dependency	functional dependency	**transitive dependency**
56	Lossless decomposition is_____	Reversible	Irreversible	Either a) or b)	None of the above	**Reversible**
57	_____ is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.	SQL Outer Join	SQL Inner Join	Self Join	Non Equi Join	**SQL Inner Join**

58	_____ is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.	Functional Dependencies	Normalization	Set Difference	Cartician Product	**Normalization**
59	_____ensure that there are no repeating groups of data.	1NF	2NF	3NF	4NF	**1NF**
60	____ attributes cannot be further divided into smaller components.	Composite	Simple / atomic	Derived	Stored	**Simple / atomic**
61	_____keyword is used to eliminates the duplicates	Union	Union all	Intersect all	Except all	**Union**
62	_____is a condition specified on a database schema & restricts the data that can be stored in an instance of the database.	integrity Constraint	restriction	key	none	**integrity Constraint**

UNIT-IV

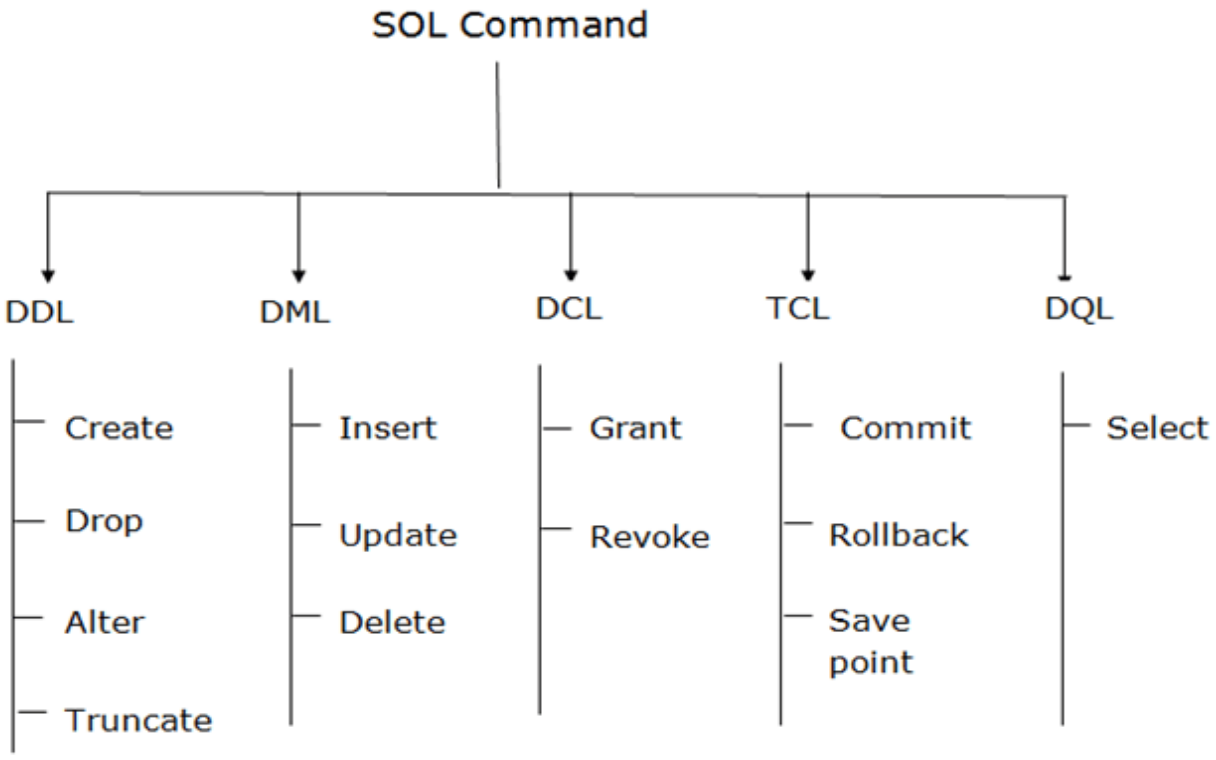
Introduction to SQL: DDL, DML, and DCL statements, Creating Tables, Adding Constraints, Altering Tables, Update, Insert, Delete & various Form of SELECT- Simple, Using Special Operators for Data Access. Aggregate functions, Joining Multiple Tables (Equi Joins),Joining a Table to itself (self Joins) Functions.

Introduction to PL/SQL: Declaration section – executable command section : conditional logic, loops, CASE statements –

4.1 SQL command

- 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 Command



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:

- CREATE
- ALTER
- DROP
- 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 ;**

   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 MODIFY(COLUMN DEFINITION....);

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:

- INSERT
- UPDATE
- 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:

- Grant
- 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:

- COMMIT
- ROLLBACK
- SAVEPOINT

a. Commit: Commit command is used to save all the transactions to the database.

Syntax:

```
1. COMMIT;
```

Example:

```
1. DELETE FROM CUSTOMERS  WHERE AGE = 25;
2. COMMIT;
```

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

```
1. ROLLBACK;
```

Example:

```
1. DELETE FROM CUSTOMERS  WHERE AGE = 25;
2. 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;
```

4.2 SQL SELECT Statement

- The SELECT statement retrieves data from a database.
- The data is returned in a table-like structure called a result-set.
- SELECT is the most frequently used action on a database.

The SQL SELECT syntax

The general syntax is:

```
1. SELECT column-names
2.   FROM table-name
```

To select all columns use *

```
1. SELECT *
2.   FROM table-name
```

Problem: List all customers

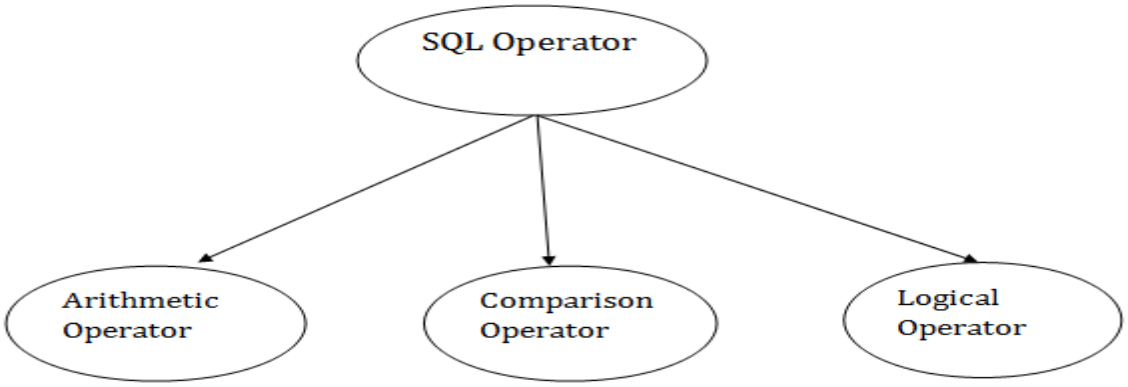
```
1. SELECT *   FROM Customer
```

d	FirstName	LastName	City	Country	Phone
1	Maria	Anders	Berlin	Germany	030-0074321
2	Ana	Trujillo	México D.F.	Mexico	(5) 555-4729
3	Antonio	Moreno	México D.F.	Mexico	(5) 555-3932
4	Thomas	Hardy	London	UK	(171) 555-7788
5	Christina	Berglund	Luleå	Sweden	0921-12 34 65

1. **SELECT FirstName, LastName, City FROM Customer**

SQL Operator

There are various types of SQL operator:



4.3 SQL Arithmetic Operators

Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.

Operator	Description	Example
+	It adds the value of both operands.	a+b will give 30
-	It is used to subtract the right-hand operand from the left-hand operand.	a-b will give 10
*	It is used to multiply the value of both operands.	a*b will give 200
/	It is used to divide the left-hand operand by the right-hand operand.	a/b will give 2
%	It is used to divide the left-hand operand by the right-hand operand and returns reminder.	a%b will give 0

4. 4 SQL Comparison Operators:

Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.

Operator	Description	Example
=	It checks if two operands values are equal or not, if the values are queal then condition becomes true.	(a=b) is not true
!=	It checks if two operands values are equal or not, if values are not equal, then condition becomes true.	(a!=b) is true
<>	It checks if two operands values are equal or not, if values are not equal then condition becomes true.	(a<>b) is true
>	It checks if the left operand value is greater than right operand	(a>b) is not

	value, if yes then condition becomes true.	true
<	It checks if the left operand value is less than right operand value, if yes then condition becomes true.	(a<b) is true
>=	It checks if the left operand value is greater than or equal to the right operand value, if yes then condition becomes true.	(a>=b) is not true
<=	It checks if the left operand value is less than or equal to the right operand value, if yes then condition becomes true.	(a<=b) is true
!<	It checks if the left operand value is not less than the right operand value, if yes then condition becomes true.	(a!=b) is not true
!>	It checks if the left operand value is not greater than the right operand value, if yes then condition becomes true.	(a!>b) is true

SQL Logical Operators

There is the list of logical operator used in SQL:

Operator	Description
ALL	It compares a value to all values in another value set.
AND	It allows the existence of multiple conditions in an SQL statement.
ANY	It compares the values in the list according to the condition.
BETWEEN	It is used to search for values that are within a set of values.
IN	It compares a value to that specified list value.
NOT	It reverses the meaning of any logical operator.
OR	It combines multiple conditions in SQL statements.
EXISTS	It is used to search for the presence of a row in a specified table.

LIKE	It compares a value to similar values using wildcard operator.

4.5 SQL Table

- SQL Table is a collection of data which is organized in terms of rows and columns. In DBMS, the table is known as relation and row as a tuple.
- Table is a simple form of data storage. A table is also considered as a convenient representation of relations.

Let's see an example of the **EMPLOYEE** table:

EMP_ID	EMP_NAME	CITY	PHONE_NO
1	Kristen	Washington	7289201223
2	Anna	Franklin	9378282882
3	Jackson	Bristol	9264783838
4	Kellan	California	7254728346
5	Ashley	Hawaii	9638482678

In the above table, "EMPLOYEE" is the table name, "EMP_ID", "EMP_NAME", "CITY",

"PHONE_NO" are the column names. The combination of data of multiple columns forms a row, e.g., 1, "Kristen", "Washington" and 7289201223 are the data of one row.

Operation on Table

1. Create table
2. Drop table
3. Delete table
4. Rename table

SQL Create Table

SQL create table is used to create a table in the database. To define the table, you should define the name of the table and also define its columns and column's data type.

1. create table "table_name"
2. ("column1" "data type",
3. "column2" "data type",
4. "column3" "data type",
5. ...
6. "columnN" "data type");

```
SQL> CREATE TABLE EMPLOYEE (
EMP_ID INT                    NOT NULL,
EMP_NAME VARCHAR (25) NOT NULL,
PHONE_NO INT                  NOT NULL,
ADDRESS CHAR (30),
PRIMARY KEY (ID));
```

Drop table

A SQL drop table is used to delete a table definition and all the data from a table. When this command is executed, all the information available in the table is lost forever, so you have to very careful while using this command.

Syntax

1. DROP TABLE "table_name";
2. SQL>DROP TABLE EMPLOYEE;

SQL DELETE table

In SQL, DELETE statement is used to delete rows from a table. We can use WHERE condition to delete a specific row from a table. If you want to delete all the records from the table, then you don't need to use the WHERE clause.

Syntax

1. DELETE FROM table_name WHERE condition;

Suppose, the EMPLOYEE table having the following records:

EMP_ID	EMP_NAME	CITY	PHONE_NO	SALARY
1	Kristen	Chicago	9737287378	150000
2	Russell	Austin	9262738271	200000
3	Denzel	Boston	7353662627	100000
4	Angelina	Denver	9232673822	600000
5	Robert	Washington	9367238263	350000
6	Christian	Los angels	7253847382	260000

The following query will DELETE an employee whose ID is 2.

```
1. SQL> DELETE FROM EMPLOYEE
2. WHERE EMP_ID = 3;
```

Now, the EMPLOYEE table would have the following records.

EMP_ID	EMP_NAME	CITY	PHONE_NO	SALARY
1	Kristen	Chicago	9737287378	150000
2	Russell	Austin	9262738271	200000
4	Angelina	Denver	9232673822	600000
5	Robert	Washington	9367238263	350000

6	Christian	Los angels	7253847382	260000

If you don't specify the WHERE condition, it will remove all the rows from the table.

1. DELETE FROM EMPLOYEE;

SQL SELECT Statement

In SQL, the SELECT statement is used to query or retrieve data from a table in the database. The returns data is stored in a table, and the result table is known as result-set.

Syntax

1. SELECT column1, column2, ...
2. FROM table_name;

Here, the expression is the field name of the table that you want to select data from.

Use the following syntax to select all the fields available in the table:

1. SELECT * FROM table_name;

Example:

EMPLOYEE

EMP_ID	EMP_NAME	CITY	PHONE_NO	SALARY
1	Kristen	Chicago	9737287378	150000
2	Russell	Austin	9262738271	200000
3	Angelina	Denver	9232673822	600000
4	Robert	Washington	9367238263	350000
5	Christian	Los angels	7253847382	260000

To fetch the EMP_ID of all the employees, use the following query:

1. SELECT EMP_ID FROM EMPLOYEE;

Output

EMP_ID
1
2
3
4
5

To fetch the EMP_NAME and SALARY, use the following query:

1. SELECT EMP_NAME, SALARY FROM EMPLOYEE;

EMP_NAME	SALARY
Kristen	150000
Russell	200000
Angelina	600000
Robert	350000
Christian	260000

To fetch all the fields from the EMPLOYEE table, use the following query:

1. SELECT * FROM EMPLOYEE

Output

EMP_ID	EMP_NAME	CITY	PHONE_NO	SALARY
1	Kristen	Chicago	9737287378	150000
2	Russell	Austin	9262738271	200000
3	Angelina	Denver	9232673822	600000
4	Robert	Washington	9367238263	350000
5	Christian	Los angels	7253847382	260000

SQL INSERT Statement

1. Without specifying column name
2. By specifying column name

1. Without specifying column name

Syntax

1. INSERT INTO TABLE_NAME
2. VALUES (value1, value2, value 3, ... Value N);

Query

1. INSERT INTO EMPLOYEE VALUES (6, 'Marry', 'Canada', 600000, 48);

Output: After executing this query, the EMPLOYEE table will look like:

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

2. By specifying column name

To insert partial column values, you must have to specify the column names.

Syntax

1. INSERT INTO TABLE_NAME
2. [(col1, col2, col3,.... col N)]
3. VALUES (value1, value2, value 3, ... Value N);

Query

1. INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, AGE) VALUES (7, 'Jack', 40);

Output: After executing this query, the table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29

5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48
7	Jack	null	null	40

SQL Update Statement

The SQL UPDATE statement is used to modify the data that is already in the database. The condition in the WHERE clause decides that which row is to be updated.

Syntax

1. UPDATE table_name
2. SET column1 = value1, column2 = value2, ...
3. WHERE condition;

Sample Table

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36

1. Without specifying column name

If you want to specify all column values, you can specify or ignore the column values.

Syntax

1. INSERT INTO TABLE_NAME
2. VALUES (value1, value2, value 3, ... Value N);

Query

1. INSERT INTO EMPLOYEE VALUES (6, 'Marry', 'Canada', 600000, 48);

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

2. By specifying column name

To insert partial column values, you must have to specify the column names.

Syntax

1. INSERT INTO TABLE_NAME
2. [(col1, col2, col3,.... col N)]
3. VALUES (value1, value2, value 3, ... Value N);

Query

1. INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, AGE) VALUES (7, 'Jack', 40);

Output: After executing this query, the table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30

2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48
7	Jack	null	null	40

Without use of WHERE clause

If you want to update all row from a table, then you don't need to use the WHERE clause. In the EMPLOYEE table, update the column EMP_NAME as 'Harry'.

Syntax

```
1. UPDATE table_name
2. SET column_name = value1;
```

Query

```
1. UPDATE EMPLOYEE
2. SET EMP_NAME = 'Harry';
```

Output

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Harry	Chicago	200000	30
2	Harry	Austin	300000	26
3	Harry	Denver	100000	42
4	Harry	Washington	500000	29

5	Harry	Los angels	200000	36
6	Harry	Canada	600000	48

SQL DELETE Statement

The SQL DELETE statement is used to delete rows from a table. Generally, DELETE statement removes one or more records form a table.

Syntax

1. DELETE FROM table_name WHERE some_condition;

Sample Table

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

Deleting Single Record

Delete the row from the table EMPLOYEE where EMP_NAME = 'Kristen'. This will delete only the fourth row.

Query

1. DELETE FROM EMPLOYEE
2. WHERE EMP_NAME = 'Kristen';

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

Deleting Multiple Record

Delete the row from the EMPLOYEE table where AGE is 30. This will delete two rows(first and third row).

Query

1. DELETE FROM EMPLOYEE WHERE AGE= 30;

Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

Delete all of the records

Delete all the row from the EMPLOYEE table. After this, no records left to display. The EMPLOYEE table will become empty.

Syntax

1. DELETE * FROM table_name;
2. or
3. DELETE FROM table_name;

Query

1. **DELETE FROM EMPLOYEE;**

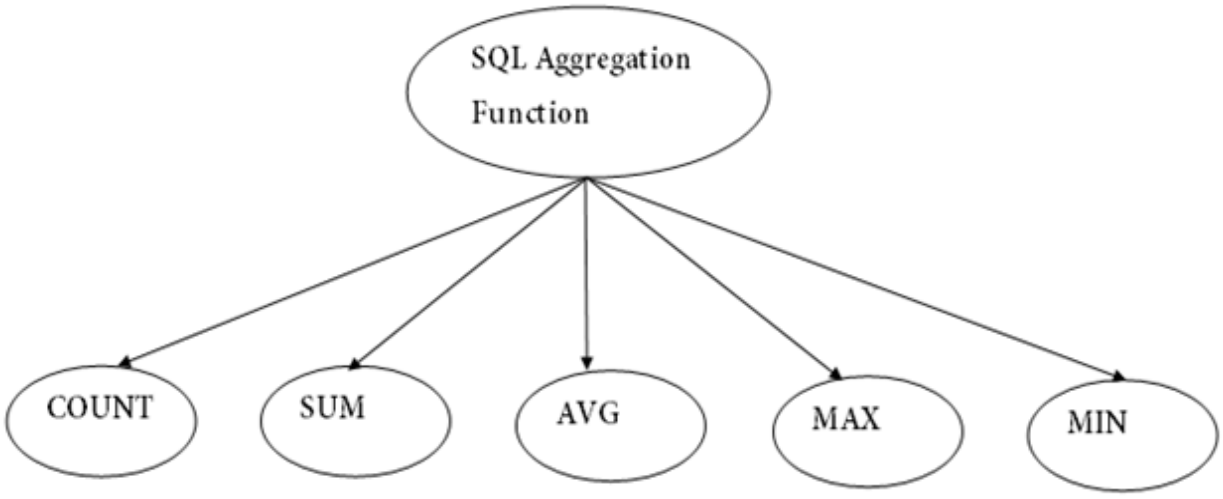
Output: After executing this query, the EMPLOYEE table will look like:

EMP_ID	EMP_NAME	CITY	SALARY	AGE

4.6 SQL Aggregate Functions

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

1. COUNT(*)
2. or
3. 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()

1. SELECT COUNT(*) FROM PRODUCT_MAST;

Output:

```
10
```

Example: COUNT with WHERE

1.
2. SELECT COUNT(*) FROM PRODUCT_MAST WHERE RATE>=20;

Output:

```
7
```

Example: COUNT() with DISTINCT

1. SELECT COUNT(DISTINCT COMPANY) FROM PRODUCT_MAST;

Output:

```
3
```

Example: COUNT() with GROUP BY

1. SELECT COMPANY, COUNT(*)
2. FROM PRODUCT_MAST GROUP BY COMPANY;

Output:

```
Com1    5
Com2    3
Com3    2
```

Example: COUNT() with HAVING

1. 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

1. SUM()
2. or
3. SUM([ALL|DISTINCT] expression)

Example: SUM()

1. SELECT SUM(COST) FROM PRODUCT_MAST;

Output:

670

Example: SUM() with WHERE

1. SELECT SUM(COST) FROM PRODUCT_MAST WHERE QTY>3;

Output:

320

Example: SUM() with GROUP BY

1. SELECT SUM(COST) FROM PRODUCT_MAST WHERE QTY>3
2. GROUP BY COMPANY;

Output:

Com1 150

Com2 170

Example: SUM() with HAVING

1. 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

1. AVG()
2. or
3. AVG([ALL|DISTINCT] expression)

Example:

1. SELECT AVG(COST) FROM PRODUCT_MAST;
2. **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

1. MAX()
2. or
3. MAX([ALL|DISTINCT] expression)

Example:

1. 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

1. MIN()
2. or
3. MIN([ALL|DISTINCT] expression)

Example:

1. SELECT MIN(RATE) FROM PRODUCT_MAST;

Output:

10

4.7 SQL JOIN

As the name shows, JOIN means to combine something. In case of SQL, JOIN means "to combine two or more tables".

In SQL, JOIN clause is used to combine the records from two or more tables in a database.

Types of SQL JOIN

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. FULL JOIN

Sample Table

EMPLOYEE

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

PROJECT

PROJECT_NO	EMP_ID	DEPARTMENT
101	1	Testing
102	2	Development
103	3	Designing
104	4	Development

1. INNER JOIN

In SQL, INNER JOIN selects records that have matching values in both tables as long as the condition is satisfied. It returns the combination of all rows from both the tables where the condition satisfies.

Syntax

```
1. SELECT table1.column1, table1.column2, table2.column1,....
2. FROM table1
3. INNER JOIN table2
4. ON table1.matching_column = table2.matching_column;
```

Query

```
1. SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT   FROM EMPLOYEE
   INNER JOIN PROJECT  ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing

Kristen	Development

2. LEFT JOIN

The SQL left join returns all the values from left table and the matching values from the right table.

If there is no matching join value, it will return NULL.

Syntax

```
1. SELECT table1.column1, table1.column2, table2.column1,....
2. FROM table1
3. LEFT JOIN table2
4. ON table1.matching_column = table2.matching_column;
```

Query

```
1. SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT   FROM EMPLOYEE
   LEFT JOIN PROJECT  ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development
Russell	NULL
Marry	NULL

3. RIGHT JOIN

In SQL, RIGHT JOIN returns all the values from the values from the rows of right table and the matched values from the left table. If there is no matching in both tables, it will return NULL.

Syntax

```
1. SELECT table1.column1, table1.column2, table2.column1,....
2. FROM table1
3. RIGHT JOIN table2
4. ON table1.matching_column = table2.matching_column;
```

Query

```
1. SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT   FROM EMPLOYEE
   RIGHT JOIN PROJECT  ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development

4. FULL JOIN

In SQL, FULL JOIN is the result of a combination of both left and right outer join. Join tables have all the records from both tables. It puts NULL on the place of matches not found.

Syntax

```
1. SELECT table1.column1, table1.column2, table2.column1,....
2. FROM table1
3. FULL JOIN table2
4. ON table1.matching_column = table2.matching_column;
```

Query

```
1. SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT FROM EMPLOYEE
```

```
FULL JOIN PROJECT   ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development
Russell	NULL
Marry	NULL

4.8 PL/SQL Tutorial

PL/SQL is Oracle's procedural language (PL) superset of the Structured Query Language

PL/SQL is a block structured language that can have multiple blocks in it.

Contain any number of nested sub-blocks. Pl/SQL stands for "Procedural Language extension of SQL" that is used in Oracle. PL/SQL is integrated with Oracle database (since version 7).

The functionalities of PL/SQL usually extended after each release of Oracle database. Although PL/SQL is closely integrated with SQL language, yet it adds some programming constraints that are not available in SQL.

PL/SQL Overview

PL/SQL code is grouped into structures called blocks. If you create a stored procedure or package, you give the block of PL/SQL code a name; if the block of PL/SQL code is not given a name, then it is called an anonymous block. A block of PL/SQL code contains three sections:

Section	Description	inclusion
Declarative	Contains all variables, constants, cursors, and user-defined exceptions that are referenced in the executable and declarative sections	Optional
Executable	Contains SQL statements to manipulate data in the database and PL/SQL statements to manipulate data in the block	Mandatory
Exception handling	Specifies the actions to perform when errors and abnormal conditions arise in the executable section	Optional

The Declarations section starts with the keyword **declare** and ends when the Executable Commands section starts (as indicated by the keyword **begin**). The Executable Commands section is followed by the Exception Handling section; the **exception** keyword signals the start of the Exception Handling section. The PL/SQL block is terminated by the **end** keyword.

The structure of a typical PL/SQL block is shown in the following listing:

declare
<declarations section>
begin
<executable commands>
exception
<exception handling>
end;

- Any block may contain sub-block. Sub-blocks may appear anywhere an executable statement may legally appear.
- Statement end with a ;
- Comments are preceded by --or surrounded by /* */ Declared objects exist within a certain scope (addressed later in this course).

4.9 PL/SQL Variables

A variable is a meaningful name which facilitates a programmer to store data temporarily during the

execution of code. It helps you to manipulate data in PL/SQL programs. It is nothing except a name given to a storage area. Each variable in the PL/SQL has a specific data type which defines the size and layout of the variable's memory.

A variable should not exceed 30 characters. Its letter optionally followed by more letters, dollar signs, numerals, underscore etc.

How to declare variable in PL/SQL

You must declare the PL/SQL variable in the declaration section or in a package as a global variable. After the declaration, PL/SQL allocates memory for the variable's value and the storage location is identified by the variable name.

Syntax for declaring variable:

Following is the syntax for declaring variable:

1. variable_name [CONSTANT] datatype [NOT NULL] [:= | **DEFAULT** initial_value]

Here, variable_name is a valid identifier in PL/SQL and datatype must be valid PL/SQL data type. A data type with size, scale or precision limit is called a constrained declaration. The constrained declaration needs less memory than unconstrained declaration.

Naming rules for PL/SQL variables

The variable in PL/SQL must follow some naming rules like other programming languages.

- The variable_name should not exceed 30 characters.
- The name of the variable must begin with ASCII letter. The PL/SQL is not case sensitive so it could be either lowercase or uppercase. For example: v_data and V_DATA refer to the same variables.
- You should make your variable easy to read and understand, after the first character, it may be any number, underscore (_) or dollar sign ($).
- NOT NULL is an optional specification on the variable.

Initializing Variables in PL/SQL

Evertime you declare a variable, PL/SQL defines a default value NULL to it. If you want to initialize a variable with other value than NULL value, you can do so during the declaration, by using any one of the following methods.

- The DEFAULT keyword
- The assignment operator

```
1. counter binary_integer := 0;
2. greetings varchar2(20) DEFAULT 'Hello JavaTpoint';
```

You can also specify NOT NULL constraint to avoid NULL value. If you specify the NOT NULL constraint, you must assign an initial value for that variable.

You must have a good programming skill to initialize variable properly otherwise, sometimes program would produce unexpected result.

Example of initilizing variable

Let's take a simple example to explain it well:

```
1.  DECLARE
2.    a integer := 30;
3.    b integer := 40;
4.    c integer;
5.    f real;
6.  BEGIN
7.    c := a + b;
8.    dbms_output.put_line('Value of c: ' || c);
9.    f := 100.0/3.0;
10.   dbms_output.put_line('Value of f: ' || f);
11. END;
```

After the execution, this will produce the following result:

```
1. Value of c: 70
2. Value of f: 33.333333333333333333
3.
4. PL/SQL procedure successfully completed.
```

Variable Scope in PL/SQL:

PL/SQL allows nesting of blocks. A program block can contain another inner block. If you declare a variable within an inner block, it is not accessible to an outer block. There are two types of variable scope:

- Local Variable: Local variables are the inner block variables which are not accessible to outer blocks.
- Global Variable: Global variables are declared in outermost block.

Example of Local and Global variables

Let's take an example to show the usage of Local and Global variables in its simple form:

```
1. DECLARE
2.  -- Global variables
3.    num1 number := 95;
4.    num2 number := 85;
5. BEGIN
6.    dbms_output.put_line('Outer Variable num1: ' || num1);
7.    dbms_output.put_line('Outer Variable num2: ' || num2);
8.    DECLARE
9.       -- Local variables
10.      num1 number := 195;
11.      num2 number := 185;
12.   BEGIN
13.      dbms_output.put_line('Inner Variable num1: ' || num1);
14.      dbms_output.put_line('Inner Variable num2: ' || num2);
15.   END;
16. END;
17. /
```

After the execution, this will produce the following result:

```
1. Outer Variable num1: 95
2. Outer Variable num2: 85
3. Inner Variable num1: 195
4. Inner Variable num2: 185
5.
```

6. PL/SQL **procedure** successfully completed.

4.10 PL/SQL Constants

A constant is a value used in a PL/SQL block that remains unchanged throughout the program. It is a user-defined literal value. It can be declared and used instead of actual values.

Suppose, you have to write a program which will increase the salary of the employees upto 30%, you can declare a constant and use it throughout the program. Next time if you want to increase the salary again you can change the value of constant than the actual value throughout the program.

1. constant_name CONSTANT datatype := VALUE;
   - **Constant_name:**it is the name of constant just like variable name. The constant word is a reserved word and its value does not change.
   - **VALUE:** it is a value which is assigned to a constant when it is declared. It can not be assigned later.

4.11 PL/SQL Literals

Literals are the explicit numeric, character, string or boolean values which are not represented by an identifier. For example: TRUE, NULL, etc. are all literals of type boolean. PL/SQL literals are case-sensitive. There are following kinds of literals in PL/SQL:

- Numeric Literals
- Character Literals
- String Literals
- BOOLEAN Literals
- Date and Time Literals

Example of these different types of Literals:

Literals	Examples
Numeric	75125, 3568, 33.3333333 etc.
Character	'A' '%' '9' ' ' 'z' '('

String	Hello JavaTpoint!
Boolean	TRUE, FALSE, NULL etc.
Date and Time	'26-11-2002' , '2012-10-29 12:01:01'

PL/SQL If

PL/SQL supports the programming language features like conditional statements and iterative statements. Its programming constructs are similar to how you use in programming languages like Java and C++.

Syntax for IF Statement:

There are different syntaxes for the IF-THEN-ELSE statement.

Syntax: (IF-THEN statement):

1. IF condition
2. **THEN**
3. Statement: {It **is** executed **when** condition **is true**}
4. **END** IF;

This syntax is used when you want to execute statements only when condition is TRUE.

Syntax: (IF-THEN-ELSE statement):

1. IF condition
2. **THEN**
3. {...statements **to execute when** condition **is TRUE**...}
4. **ELSE**
5. {...statements **to execute when** condition **is FALSE**...}
6. **END** IF;

This syntax is used when you want to execute one set of statements when condition is TRUE or a different set of statements when condition is FALSE.

Example of PL/SQL If Statement

Let's take an example to see the whole concept:

```
1.  DECLARE
2.    a number(3) := 500;
3.  BEGIN
4.    -- check the boolean condition using if statement
5.    IF( a < 20 ) THEN
6.       -- if condition is true then print the following
7.       dbms_output.put_line('a is less than 20 ' );
8.    ELSE
9.       dbms_output.put_line('a is not less than 20 ' );
10.   END IF;
11.   dbms_output.put_line('value of a is : ' || a);
12. END;
```

After the execution of the above code in SQL prompt, you will get the following result:

```
a is not less than 20
value of a is : 500
PL/SQL procedure successfully completed.
```

4.12. PL/SQL Case Statement

The PL/SQL CASE statement facilitates you to execute a sequence of satatements based on a selector. A selector can be anything such as variable, function or an expression that the CASE statement checks to a boolean value.

The CASE statement works like the IF statement, only using the keyword WHEN. A CASE statement is evaluated from top to bottom. If it get the condition TRUE, then the corresponding THEN calause is executed and the execution goes to the END CASE clause.

Syntax for the CASE Statement:

```
1.  CASE [ expression ]
2.  WHEN condition_1 THEN result_1
3.    WHEN condition_2 THEN result_2
4.    ...
5.    WHEN condition_n THEN result_n
6.   ELSE result
7.  END
```

Example of PL/SQL case statement

Let's take an example to make it clear:

```
1. DECLARE
2.    grade char(1) := 'A';
            1. BEGIN
            2.    CASE grade
            3.      when 'A' then dbms_output.put_line('Excellent');
            4.      when 'B' then dbms_output.put_line('Very good');
            5.      when 'C' then dbms_output.put_line('Good');
            6.      when 'D' then dbms_output.put_line('Average');
            7.      when 'F' then dbms_output.put_line('Passed with Grace');
            8.      else dbms_output.put_line('Failed');
            9.    END CASE;
            10. END;
```

After the execution of above code, you will get the following result:

```
Excellent
PL/SQL procedure successfully completed.
```

PL/SQL Loop

The PL/SQL loops are used to repeat the execution of one or more statements for specified number of times. These are also known as iterative control statements.

Syntax for a basic loop:

```
1. LOOP
2.   Sequence of statements;
3. END LOOP;
```

Types of PL/SQL Loops

There are 4 types of PL/SQL Loops.

1. Basic Loop / Exit Loop
2. While Loop
3. For Loop

4. Cursor For Loop

PL/SQL Exit Loop (Basic Loop)

PL/SQL exit loop is used when a set of statements is to be executed at least once before the termination of the loop. There must be an EXIT condition specified in the loop, otherwise the loop will get into an infinite number of iterations. After the occurrence of EXIT condition, the process exits the loop.

Syntax of basic loop:

```
1. LOOP
2.   Sequence of statements;
3. END LOOP;
```

Syntax of exit loop:

```
1. LOOP
2.   statements;
3.   EXIT;
4.   {or EXIT WHEN condition;}
5. END LOOP;
```

Example of PL/SQL EXIT Loop

Let's take a simple example to explain it well:

```
1. DECLARE
2. i NUMBER := 1;
3. BEGIN
4. LOOP
5. EXIT WHEN i>10;
6. DBMS_OUTPUT.PUT_LINE(i);
7. i := i+1;
8. END LOOP;
9. END;
```

After the execution of the above code, you will get the following result:

```
1
```

```
2
3
4
5
6
7
8
9
10
```

Note: You must follow these steps while using PL/SQL Exit Loop.

- Initialize a variable before the loop body
- Increment the variable in the loop.
- You should use EXIT WHEN statement to exit from the Loop. Otherwise the EXIT statement without WHEN condition, the statements in the Loop is executed only once.

PL/SQL EXIT Loop Example 2

```
1. DECLARE
2. VAR1 NUMBER;
3. VAR2 NUMBER;
4. BEGIN
5. VAR1:=100;
6. VAR2:=1;
7. LOOP
8. DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
9. IF (VAR2=10) THEN
10. EXIT;
11. END IF;
12. VAR2:=VAR2+1;
13. END LOOP;
14. END;
```

Output:

```
100
200
300
400
500
600
```

```
700
800
900
1000
```

4.13 PL/SQL While Loop

PL/SQL while loop is used when a set of statements has to be executed as long as a

condition is true, the While loop is used. The condition is decided at the beginning of each iteration and continues until the condition becomes false.

Syntax of while loop:

```
1. WHILE <condition>
2.  LOOP statements;
3. END LOOP;
```

Example of PL/SQL While Loop

Let's see a simple example of PL/SQL WHILE loop.

```
1. DECLARE
2. i INTEGER := 1;
3. BEGIN
4. WHILE i <= 10 LOOP
5. DBMS_OUTPUT.PUT_LINE(i);
6. i := i+1;
7. END LOOP;
8. END;
```

After the execution of the above code, you will get the following result:

```
1
2
3
4
5
6
7
8
9
10
```

Note: You must follow these steps while using PL/SQL WHILE Loop.

- Initialize a variable before the loop body.
- Increment the variable in the loop.
- You can use EXIT WHEN statements and EXIT statements in While loop but it is not done often.

PL/SQL WHILE Loop Example 2

```
1. DECLARE
2. VAR1 NUMBER;
3. VAR2 NUMBER;
4. BEGIN
5. VAR1:=200;
6. VAR2:=1;
7. WHILE (VAR2<=10)
8. LOOP
9. DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
10. VAR2:=VAR2+1;
11. END LOOP;
12. END;
```

Output:

```
200
400
600
800
1000
1200
1400
1600
1800
2000
```

4.15 PL/SQL FOR Loop

PL/SQL for loop is used when when you want to execute a set of statements for a predetermined number of times. The loop is iterated between the start and end integer values. The counter is always incremented by 1 and once the counter reaches the value of end integer, the loop ends.

Syntax of for loop:

1. **FOR** counter IN initial_value .. final_value LOOP
2. LOOP statements;
3. **END** LOOP;
    - initial_value : Start integer value
    - final_value : End integer value

PL/SQL For Loop Example 1

Let's see a simple example of PL/SQL FOR loop.

```
1. BEGIN
2. FOR k IN 1..10 LOOP
3. -- note that k was not declared
4. DBMS_OUTPUT.PUT_LINE(k);
5. END LOOP;
6. END;
```

After the execution of the above code, you will get the following result:

```
1
2
3
4
5
6
7
8
9
10
```

Note: You must follow these steps while using PL/SQL WHILE Loop.

- You don't need to declare the counter variable explicitly because it is declared implicitly in the declaration section.
- The counter variable is incremented by 1 and does not need to be incremented explicitly.
- You can use EXIT WHEN statements and EXIT statements in FOR Loops but it is not done often.

PL/SQL For Loop Example 2

```
1. DECLARE
```

```
2. VAR1 NUMBER;
3. BEGIN
4. VAR1:=10;
5. FOR VAR2 IN 1..10
6. LOOP
7. DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
8. END LOOP;
9. END;
```

Output:

```
10
20
30
40
50
60
70
80
90
100
```

PL/SQL For Loop REVERSE Example 3

Let's see an example of PL/SQL for loop where we are using REVERSE keyword.

```
1. DECLARE
2. VAR1 NUMBER;
3. BEGIN
4. VAR1:=10;
5. FOR VAR2 IN REVERSE 1..10
6. LOOP
7. DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
8. END LOOP;
9. END;
```

Output:

```
100
90
80
70
```

```
60
50
40
30
20
10
```

4.16 PL/SQL Continue Statement

The continue statement is used to exit the loop from the reminder if its body either conditionally or unconditionally and forces the next iteration of the loop to take place, skipping any codes in between.

The continue statement is not a keyword in Oracle 10g. It is a new feature encorporated in oracle 11g.

For example: If a continue statement exits a cursor FOR LOOP prematurely then it exits an inner loop and transfer control to the next iteration of an outer loop, the cursor closes (in this context, CONTINUE works like GOTO).

Syntax:

1. **continue**;

   Example of PL/SQL continue statement

   Let's take an example of PL/SQL continue statement.

```
1.  DECLARE
2.    x NUMBER := 0;
3.  BEGIN
4.    LOOP -- After CONTINUE statement, control resumes here
5.      DBMS_OUTPUT.PUT_LINE ('Inside loop:  x = ' || TO_CHAR(x));
6.      x := x + 1;
7.      IF x < 3 THEN
8.        CONTINUE;
9.      END IF;
10.     DBMS_OUTPUT.PUT_LINE
11.       ('Inside loop, after CONTINUE:  x = ' || TO_CHAR(x));
12.     EXIT WHEN x = 5;
13.   END LOOP;
14.
```

```
15.  DBMS_OUTPUT.PUT_LINE (' After loop:  x = ' || TO_CHAR(x));
16. END;
17. /
```

After the execution of above code, you will get the following result:

```
Inside loop:  x = 0
Inside loop:  x = 1
Inside loop:  x = 2
Inside loop, after CONTINUE:  x = 3
Inside loop:  x = 3
Inside loop, after CONTINUE:  x = 4
Inside loop:  x = 4
Inside loop, after CONTINUE:  x = 5
After loop:  x = 5
```

PL/SQL GOTO Statement

Syntax:

1. **GOTO** label_name;

Here the label declaration which contains the label_name encapsulated within the << >> symbol and must be followed by at least one statement to execute.

```
1. GOTO label_name;
2.  ..
3. ..
4. <<label_name>>
5. Statement;
```

Example of PL/SQL GOTO statement

Let's take an example of PL/SQL GOTO statement.

```
1. DECLARE
2.    a number(2) := 30;
3. BEGIN
4.    <<loopstart>>
5.    -- while loop execution
        1.    WHILE a < 50 LOOP
        2.       dbms_output.put_line ('value of a: ' || a);
```

```
3.       a := a + 1;
4.       IF a = 35 THEN
5.          a := a + 1;
6.          GOTO loopstart;
7.       END IF;
8.    END LOOP;
9.  END;
10. /
```

After the execution of above code, you will get the following result:

```
value of a: 30
value of a: 31
value of a: 32
value of a: 33
value of a: 34
value of a: 36
value of a: 37
value of a: 38
value of a: 39
value of a: 40
value of a: 41
value of a: 42
value of a: 43
value of a: 44
value of a: 45
value of a: 46
value of a: 47
value of a: 48
value of a: 49

Statement processed.
```

Part-A

(Online – Multiple choice questions)

(Each questions carries one mark each)

PART – B (2 Marks)

1. Write a short note on any one of the conditional logic command.
2. What are cursors?
3. What is user defined exception?
4. What are data types?
5. What are triggers and give its types?

PART – C (8 Marks)

1. What are cursors? Discuss how you will use cursors in PL/SQL with syntax.
2. What are Triggers. How are triggers used in PL/SQL. Explain with syntax
3. What are CASE expressions? Give the syntax and write a PL/SQL block to print Grade of a student given the mark percentage.
4. Write in detail about i) Internal exception ii) User defined exceptions
5. Write in detail about various data types used in PL/SQL.
6. Describe the characteristics of using functions in PL/SQL. Give the syntax for defining a function.
7. Write in detail about the conditional structures in PL/SQL. Give the syntax and explain with an example.
8. What are procedures? With a detailed explanation give the structure of a procedure with example.
9. Describe the looping statements used in PL/SQL with example.

Part – A - Included in Excel file – **File name** Unit-IV(MCQ).xls

- Objective type / Multiple choice questions. Each question carries one mark
- It is for online examination

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

Subject: Relational Database Management System Sub.code: 18CTU303

S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____clause is used to group the values under particular characteristics	Order By	Group by	Count	Sum	**Group by**
2	Group function is otherwise known as_____	Collection	Aggregate	function	Count	**Aggregate**
3	_____keyword is used to assign a default value with a domain.	Static	Default	Permanent	distinct	**Default**
4	_____is a procedural language extension to SQL.	PSQL	PL/SQL	PRO/SQL	RA	**PL/SQL**
5	_____is a superset of SQL.	PSQL	PL/SQL	PRO/SQL	RA	**PL/SQL**
6	code block starts with _____section in PL/SQL	begin	end	start	declare	**declare**
7	_____ changes the structure of the table like creating a table, deleting a table, altering	DML	DDL	TCL	DCL	**DDL**
8	_____is a numeric value or a character string used to represent itself.	character set	identifiers	Lexical Units	literals	**literals**
9	_____literal returns integers or floats.	boolean	numeric	decimal	character	**numeric**

10	_____are predetermined constants	logical literal	numeric literal	float literal	character literal	**float literal**
11	_____is used to declare the variable in the PL/SQL.	character literal	numeric literal	logical literal	identifiers	**logical literal**
12	_____can not be used as a variable names.	words	literals	numbers	reserved words	**reserved words**
13	_____data type is used to store binary data	varchar2	char	raw	date	**raw**
14	The _____section marks the end of a PL/SQL block.	Terminate	end	last	next	**end**
15	The maximum length of the long raw column is_____	2GB	4GB	6GB	1GB	**2GB**
16	_____statement is used to define a new table.	Create	Produce	Insert	Add	**Create**
17	Tuples are inserted using the _____command	Create	Insert	Add	Make	**Insert**
18	Tuples are deleted using the _____command	Delete	drop	remove	alter	**Delete**
19	Modify the column values in an existing row using_____ command	Modify	Alter	Update	Change	**Update**
20	_____clause is used to modify a particular row.	Update	Modify	Alter	Where	**Update**
21	_____is used to calculate the number of values in the Column.	Count	aggregate	Cal	Calculate	**Count**
22	_____is used to calculate the sum of all values in the column	Total	Sum	Count	Collection	**Sum**
23	_____is used to calculate the average of all values in the column	Total	Sum	Average	avg	**avg**

24	Which function is used to extract the maximum value in the relations?	max	maximum	excess	large	**max**
25	Which function is used to extract the minimum value in the relations?	Small	minimum	lower	min	**min**
26	Which clause is used in conjuction with Group by clause to check a condition?	where	Having	Distinct	Group	**Having**
27	_____keyword is used to eliminates the duplicates	Union	Union all	Intersect all	Except all	**Union**
28	_____keyword is used to retain the duplicates	Union	Union all	Intersect	Except	**Union all**
29	_____is a query that has another query embedded within it.	Query	nested query	QBE	QUEL	**nested query**
30	Embedded query is called_____	host language	embedded language	subquery	QBE	**subquery**
31	_____ is an object used to speed up the searching process	Table	View	Sequence	Index	**Index**
32	_____ logical operator is used to search for the presence of a row in a specified table.	EXISTS	BETWEEN	ANY	IN	**EXISTS**
33	Duplicates are eliminated by using _____ keyword.	Remove	Distinct	RM	Redundant	**Distinct**
34	Which keyword is used to check if an element is present a given set?	not	in	not exist	except	**in**
35	Any two tables that are Union-Compatible that is, have the same number of _____	Columns	rows	Columns and rows	null values	**Columns**

36	_____is used when the column value is either unknown or inapplicable.	zero	all	Empty	null	**null**
37	___ are used to relate information in different tables	Join	Selection	Projection	Intersection	**Join**
38	_____ join condition makes use of comparison operators other than the equal sign	Inner join	Outer join	Equi join	Non equi join	**Non equi join**
39	The + symbol is used on one side of the join condition in ____ join	Inner join	Outer join	Equi join	Non equi join	**Outer join**
40	___ is a type of sql join which is used to join a table to itself	Inner join	Outer join	Equi join	Self join	**Self join**
41	____ keyword is used to define an alias name	in	from	as	to	**as**
42	A _____ is the person who accepts the privileges provided to him	Granter	Grantee	Administrator	Revoker	**Grantee**
43	_____ keyword is used to provide a specific privilege to all the users of database.	Public	All	All Users	Public users	**Public**
44	____ the command used to withdraw the access privilege given already	Withdraw	Revoke	Remove	Extract	**Revoke**
45	____ the command used to provide access privilege to the user.	Withdraw	Revoke	Grant	Allow	**Grant**
46	How many forms of Insert statement is provided by SQL?	2	3	4	5	**3**
47	The following is not a TCL command.	savepoint	rollback	commit	revoke	**revoke**

48	_____ is used to delete all the rows from the table and free the space containing the table.	terminate	delete	drop	truncate	**truncate**
49	_____ command is used to delete an index	Delete	Remove	Drop	Delete all	**Drop**
50	____ holds a string with a maximum length of 65,535 characters	LONGTEXT	MEDIUMTEXT	TINYTEXT	TEXT	**TEXT**
51	In _____ datatype if more than 255 characters are stored it will be converted to TEXT type	CHAR	VARCHAR	TINYTEXT	TEXT	**VARCHAR**
52	_____is used to delete both the structure and record stored in the table.	TRUNCATE	DELETE	ALTER	DROP	**DROP**
53	If the block of PL/SQL code is not given a name, then it is called _____	anonymous block	null block	empty block	fuctional block	**anonymous block**
54	The _____ section begins a PL/SQL block	exception	declaration	begin	end	**declaration**
55	PL/SQL _____is used when we want to execute a set of statements for a predetermined number of times.	for loop	while loop	switch case	if else	**for loop**
56	The _____statement is used to exit the loop from the reminder if its body either conditionally or unconditionally and forces the next iteration of the loop to take place, skipping any codes in between.	break	continue	go to	exit	**continue**
57	A _____ loop that repeats a specified number of times	FOR	WHILE	Simple loops	IF	**FOR**

58	An _____ clause identifies the end of the loop	exit	exit loop	end loop	end	**end loop**
59	Since the number of times the loop is executed is set when the loop is begun, an _____ command isn't needed within the loop	end	last	ending	exit	**exit**
60	The _____ function is used to calculate the average value of the numeric type.	MIN	AVG	MAX	COUNT	**AVG**
61	A _____ expression selects a result and returns it	WHILE	CASE	DO	FOR	**CASE**

UNIT-V

Exception handling section: predefined and user defined exceptions. Triggers: definition – types: row level, statement level, before and after, instead of – syntax – enabling and disabling triggers - replacing and dropping triggers. Cursors – definition – open – fetch – close – cursor attributes- select for update – types : implicit, explicit. Procedures, Functions: Local and global – procedures vs functions – stored procedures, functions – create procedure syntax - create function syntax – calling procedures, functions. Replacing and dropping procedures, functions. Package header – package body – calling package members - Replacing and dropping package.

5.0 PL/SQL Exception Handling

What is Exception?

An error occurs during the program execution is called Exception in PL/SQL.
PL/SQL facilitates programmers to catch such conditions using exception block in the program and an appropriate action is taken against the error condition.

There are two type of exceptions:

- System-defined Exceptions
- User-defined Exceptions

PL/SQL Exception Handling

Syntax for exception handling:

```
1. DECLARE
2.    <declarations section>
3. BEGIN
4.    <executable command(s)>
5. EXCEPTION
6.    <exception handling goes here >
7.    WHEN exception1 THEN
8.       exception1-handling-statements
9.    WHEN exception2  THEN
10.      exception2-handling-statements
11.   WHEN exception3 THEN
12.      exception3-handling-statements
13.   ........
14.   WHEN others THEN
15.      exception3-handling-statements
16. END;
```

Example of exception handling

SELECT* FROM COUSTOMERS;

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

```
1.  DECLARE
2.    c_id customers.id%type := 8;
3.    c_name  customers.name%type;
4.    c_addr customers.address%type;
5.  BEGIN
6.    SELECT  name, address INTO  c_name, c_addr
7.    FROM customers
8.    WHERE id = c_id;
9.  DBMS_OUTPUT.PUT_LINE ('Name: '||  c_name);
10.  DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
11. EXCEPTION
12.   WHEN no_data_found THEN
13.     dbms_output.put_line('No such customer!');
14.   WHEN others THEN
15.     dbms_output.put_line('Error!');
16. END;
17. /
```

After the execution of above code at SQL Prompt, it produces the following result:

```
No such customer!
PL/SQL procedure successfully completed.
```

The above program should show the name and address of a customer as result whose ID is given.

But there is no customer with ID value 8 in our database, so the program raises the run-time exception NO_DATA_FOUND, which is captured in EXCEPTION block.

```
1.  DECLARE
2.    c_id customers.id%type := 5;
3.    c_name  customers.name%type;
4.    c_addr customers.address%type;
5.  BEGIN
6.    SELECT  name, address INTO  c_name, c_addr
7.    FROM customers
8.    WHERE id = c_id;
9.  DBMS_OUTPUT.PUT_LINE ('Name: '||  c_name);
10.  DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
11. EXCEPTION
12.   WHEN no_data_found THEN
13.     dbms_output.put_line('No such customer!');
14.   WHEN others THEN
15.     dbms_output.put_line('Error!');
16. END;
17. /
```

After the execution of above code at SQL prompt, you will get the following result:

```
Name: alex
Address: paris
PL/SQL procedure successfully completed.
```

Raising Exceptions

In the case of any internal database error, exceptions are raised by the database server automatically But it can also be raised explicitly by programmer by using command RAISE.

Syntax for raising an exception:

```
1.  DECLARE
2.    exception_name EXCEPTION;
3.  BEGIN
4.    IF condition THEN
5.      RAISE exception_name;
```

```
6.    END IF;
7.  EXCEPTION
8.    WHEN exception_name THEN
9.    statement;
10. END;
```

5.1 PL/SQL User-defined Exceptions

PL/SQL facilitates their users to define their own exceptions according to the need of the program. A user-defined exception can be raised explicitly, using either a RAISE statement or the procedure DBMS_STANDARD.RAIS

E_APPLICATION_ERROR.

Syntax for user define exceptions

```
1. DECLARE
2. my-exception EXCEPTION;
```

PL/SQL Pre-defined Exceptions

There are many pre-defined exception in PL/SQL which are executed when any database rule is violated by the programs.
For example: NO_DATA_FOUND is a pre-defined exception which is raised when a SELECT INTO statement returns no rows.

Following is a list of some important pre-defined exceptions:

Exception	Oracle Error	SQL Code	Description
ACCESS_INTO_NULL	06530	-6530	It is raised when a NULL object is automatically assigned a value.
CASE_NOT_FOUND	06592	-6592	It is raised when none of the choices in the "WHEN" clauses of a CASE statement is selected, and there is no else clause.
COLLECTION_IS_NULL	06531	-6531	It is raised when a program attempts to apply collection methods other than exists to an uninitialized nested table or varray, or the program attempts to assign values to the

			elements of an uninitialized nested table or varray.
DUP_VAL_ON_INDEX	00001	-1	It is raised when duplicate values are attempted to be stored in a column with unique index.
INVALID_CURSOR	01001	-1001	It is raised when attempts are made to make a cursor operation that is not allowed, such as closing an unopened cursor.
INVALID_NUMBER	01722	-1722	It is raised when the conversion of a character string into a number fails because the string does not represent a valid number.
LOGIN_DENIED	01017	-1017	It is raised when s program attempts to log on to the database with an invalid username or password.
NO_DATA_FOUND	01403	+100	It is raised when a select into statement returns no rows.
NOT_LOGGED_ON	01012	-1012	It is raised when a database call is issued without being connected to the database.
PROGRAM_ERROR	06501	-6501	It is raised when PL/SQL has an internal problem.
ROWTYPE_MISMATCH	06504	-6504	It is raised when a cursor fetches value in a variable having incompatible data type.
SELF_IS_NULL	30625	-30625	It is raised when a member method is invoked, but the instance of the object type was not initialized.
STORAGE_ERROR	06500	-6500	It is raised when PL/SQL ran out of memory or memory was corrupted.
TOO_MANY_ROWS	01422	-1422	It is raised when a SELECT INTO statement returns more than one row.

VALUE_ERROR	06502	-6502	It is raised when an arithmetic, conversion, truncation, or size-constraint error occurs.
ZERO_DIVIDE	01476	1476	It is raised when an attempt is made to divide a number by zero.

5.3 PL/SQL Trigger

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

Creating a trigger:

Syntax for creating trigger:

```
1. CREATE [OR REPLACE ] TRIGGER trigger_name
2. {BEFORE | AFTER | INSTEAD OF }
3. {INSERT [OR] | UPDATE [OR] | DELETE}
4. [OF col_name]
5. ON table_name
6. [REFERENCING OLD AS o NEW AS n]
7. [FOR EACH ROW]
8. WHEN (condition)
9. DECLARE
10.    Declaration-statements
11. BEGIN
12.    Executable-statements
13. EXCEPTION
14.    Exception-handling-statements
15. 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:

```
1.  CREATE OR REPLACE TRIGGER display_salary_changes
2.  BEFORE DELETE OR INSERT OR UPDATE ON customers
3.  FOR EACH ROW
4.  WHEN (NEW.ID > 0)
5.  DECLARE
6.    sal_diff number;
7.  BEGIN
8.    sal_diff := :NEW.salary  - :OLD.salary;
9.    dbms_output.put_line('Old salary: ' || :OLD.salary);
10.   dbms_output.put_line('New salary: ' || :NEW.salary);
11.   dbms_output.put_line('Salary difference: ' || sal_diff);
12. END;
```

13. /

After the execution of the above code at SQL Prompt, it produces the following result.

```
Trigger created.
```

Check the salary difference by procedure:

Use the following code to get the old salary, new salary and salary difference after the trigger created.

```
1.  DECLARE
2.    total_rows number(2);
3.  BEGIN
4.    UPDATE  customers
5.    SET salary = salary + 5000;
6.    IF sql%notfound THEN
7.       dbms_output.put_line('no customers updated');
8.    ELSIF sql%found THEN
9.       total_rows := sql%rowcount;
10.      dbms_output.put_line( total_rows || ' customers updated ');
11.   END IF;
12. END;
13. /
```

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.

After the execution of above code again, you will get the following 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 avialable 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.

5.4 PL/SQL Cursor

When an SQL statement is processed, Oracle creates a memory area known as context area. A cursor is a pointer to this context area. It contains all information needed for processing the statement. In PL/SQL, the context area is controlled by Cursor. A cursor contains information on a select statement and the rows of data accessed by it.

A cursor is used to referred to a program to fetch and process the rows returned by the SQL

statement, one at a time. There are two types of cursors:

- Implicit Cursors
- Explicit Cursors

1) PL/SQL Implicit Cursors

The implicit cursors are automatically generated by Oracle while an SQL statement is executed,

if you don't use an explicit cursor for the statement.

These are created by default to process the statements when DML statements like INSERT, UPDATE, DELETE etc. are executed.

Orcale provides some attributes known as Implicit cursor's attributes to check the status of DML operations. Some of them are: %FOUND, %NOTFOUND, %ROWCOUNT and %ISOPEN.

For example: When you execute the SQL statements like INSERT, UPDATE, DELETE then the cursor attributes tell whether any rows are affected and how many have been affected. If you run a SELECT INTO statement in PL/SQL block, the implicit cursor attribute can be used to find out whether any row has been returned by the SELECT statement. It will return an error if there no data is selected.

The following table soecifies the status of the cursor with each of its attribute.

Attribute	Description
%FOUND	Its return value is TRUE if DML statements like INSERT, DELETE and UPDATE affect at least one row or more rows or a SELECT INTO statement returned one or more rows. Otherwise it returns FALSE.
%NOTFOUND	Its return value is TRUE if DML statements like INSERT, DELETE and

	UPDATE affect no row, or a SELECT INTO statement return no rows. Otherwise it returns FALSE. It is a just opposite of %FOUND.
%ISOPEN	It always returns FALSE for implicit cursors, because the SQL cursor is automatically closed after executing its associated SQL statements.
%ROWCOUNT	It returns the number of rows affected by DML statements like INSERT, DELETE, and UPDATE or returned by a SELECT INTO statement.

PL/SQL Implicit Cursor Example

Create customers 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

Let's execute the following program to update the table and increase salary of each customer by 5000. Here, SQL%ROWCOUNT attribute is used to determine the number of rows affected:

Create procedure:

```
1. DECLARE
2.   total_rows number(2);
3. BEGIN
4.   UPDATE  customers
5.   SET salary = salary + 5000;
```

```
6.    IF sql%notfound THEN
7.      dbms_output.put_line('no customers updated');
8.    ELSIF sql%found THEN
9.      total_rows := sql%rowcount;
10.     dbms_output.put_line( total_rows || ' customers updated ');
11.   END IF;
12. END;
13. /
```

Output:

```
6 customers updated
PL/SQL procedure successfully completed.
```

Now, if you check the records in customer table, you will find that the rows are updated.

```
1. select * from customers;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	23	Allahabad	25000
2	Suresh	22	Kanpur	27000
3	Mahesh	24	Ghaziabad	29000
4	Chandan	25	Noida	31000
5	Alex	21	Paris	33000
6	Sunita	20	Delhi	35000

2) PL/SQL Explicit Cursors

The Explicit cursors are defined by the programmers to gain more control over the context area.

These cursors should be defined in the declaration section of the PL/SQL block. It is created on a SELECT statement which returns more than one row.

Following is the syntax to create an explicit cursor:

Syntax of explicit cursor

Following is the syntax to create an explicit cursor:

1. **CURSOR** cursor_name **IS** select_statement;;

Steps:

You must follow these steps while working with an explicit cursor.

1. Declare the cursor to initialize in the memory.
2. Open the cursor to allocate memory.
3. Fetch the cursor to retrieve data.
4. Close the cursor to release allocated memory.

1) Declare the cursor:

It defines the cursor with a name and the associated SELECT statement.

Syntax for explicit cursor decleration

1. **CURSOR name IS**
2. **SELECT** statement;

2) Open the cursor:

It is used to allocate memory for the cursor and make it easy to fetch the rows returned by the SQL statements into it.

Syntax for cursor open:

1. **OPEN** cursor_name;

3) Fetch the cursor:

It is used to access one row at a time. You can fetch rows from the above-opened cursor as follows:

Syntax for cursor fetch:

1. **FETCH** cursor_name **INTO** variable_list;

4) Close the cursor:

It is used to release the allocated memory. The following syntax is used to close the above-opened cursors.

Syntax for cursor close:

1. **Close** cursor_name;

PL/SQL Explicit Cursor Example

Explicit cursors are defined by programmers to gain more control over the context area. It is defined in the declaration section of the PL/SQL block. It is created on a SELECT statement

which returns more than one row.

Let's take an example to demonstrate the use of explicit cursor. In this example, we are using the already created CUSTOMERS table.

Create customers 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 procedure:

Execute the following program to retrieve the customer name and address.

```
1. DECLARE
2.    c_id customers.id%type;
```

```
3.    c_name customers.name%type;
4.    c_addr customers.address%type;
5.    CURSOR c_customers is
6.      SELECT id, name, address FROM customers;
7.  BEGIN
8.    OPEN c_customers;
9.    LOOP
10.     FETCH c_customers into c_id, c_name, c_addr;
11.     EXIT WHEN c_customers%notfound;
12.     dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
13.   END LOOP;
14.   CLOSE c_customers;
15. END;
16. /
```

Output:

```
1  Ramesh  Allahabad
2  Suresh  Kanpur
3  Mahesh  Ghaziabad
4  Chandan  Noida
5  Alex  Paris
6  Sunita  Delhi
PL/SQL procedure successfully completed.
```

5.5 PL/SQL Function

The PL/SQL Function is very similar to PL/SQL Procedure. The main difference between procedure and a function is, a function must always return a value, and on the other hand a procedure may or may not return a value. Except this, all the other things of PL/SQL procedure are true for PL/SQL function too.

Syntax to create a function:

```
1. CREATE [OR REPLACE] FUNCTION function_name [parameters]
2. [(parameter_name [IN | OUT | IN OUT] type [, ...])]
3. RETURN return_datatype
4. {IS | AS}
5. BEGIN
```

```
6.    < function_body >
7.  END [function_name];
```

Here:

- **Function_name:** specifies the name of the function.
- **[OR REPLACE]** option allows modifying an existing function.
- The **optional parameter list** contains name, mode and types of the parameters.
- **IN** represents that value will be passed from outside and OUT represents that this parameter will be used to return a value outside of the procedure.

The function must contain a return statement.

- RETURN clause specifies that data type you are going to return from the function.
- Function_body contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

PL/SQL Function Example

Let's see a simple example to **create a function**.

```
1.  create or replace function adder(n1 in number, n2 in number)
2.  return number
3.  is
4.  n3 number(8);
5.  begin
6.  n3 :=n1+n2;
7.  return n3;
8.  end;
9.  /
```

Now write another program to **call the function**.

```
1.  DECLARE
2.    n3 number(2);
3.  BEGIN
4.    n3 := adder(11,22);
5.    dbms_output.put_line('Addition is: ' || n3);
6.  END;
7.  /
```

Output:

```
Addition is: 33
Statement processed.
0.05 seconds
```

Another PL/SQL Function Example

Let's take an example to demonstrate Declaring, Defining and Invoking a simple PL/SQL function which will compute and return the maximum of two values.

```
1.  DECLARE
2.    a number;
3.    b number;
4.    c number;
5.  FUNCTION findMax(x IN number, y IN number)
6.  RETURN number
7.  IS
8.     z number;
9.  BEGIN
10.   IF x > y THEN
11.     z:= x;
12.   ELSE
13.     Z:= y;
14.   END IF;
15.
16.   RETURN z;
17. END;
18. BEGIN
19.   a:= 23;
20.   b:= 45;
21.
22.   c := findMax(a, b);
23.   dbms_output.put_line(' Maximum of (23,45): ' || c);
24. END;
25. /
```

Output:

```
Maximum of (23,45): 45
```

```
Statement processed.
0.02 seconds
```

PL/SQL function example using table

Let's take a customer table. This example illustrates creating and calling a standalone function. This function will return the total number of CUSTOMERS in the customers table.

Customers			
Id	**Name**	**Department**	**Salary**
1	alex	web developer	35000
2	ricky	program developer	45000
3	mohan	web designer	35000
4	dilshad	database manager	44000

Create Function:

```
1. CREATE OR REPLACE FUNCTION totalCustomers
2. RETURN number IS
3.   total number(2) := 0;
4. BEGIN
5.   SELECT count(*) into total
6.   FROM customers;
7.    RETURN total;
8. END;
9. /
```

After the execution of above code, you will get the following result.

```
Function created.
```

Calling PL/SQL Function:

While creating a function, you have to give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task. Once the function is called, the program control is transferred to the called function.

After the successful completion of the defined task, the call function returns program control back to the main program.

To call a function you have to pass the required parameters along with function name and if function returns a value then you can store returned value. Following program calls the function totalCustomers from an anonymous block:

```
1. DECLARE
2.    c number(2);
3. BEGIN
4.    c := totalCustomers();
5.    dbms_output.put_line('Total no. of Customers: ' || c);
6. END;
7. /
```

After the execution of above code in SQL prompt, you will get the following result.

```
Total no. of Customers: 4
PL/SQL procedure successfully completed.
```

PL/SQL Recursive Function

You already know that a program or a subprogram can call another subprogram. When a subprogram calls itself, it is called recursive call and the process is known as recursion.

Example to calculate the factorial of a number

Let's take an example to calculate the factorial of a number. This example calculates the factorial of a given number by calling itself recursively.

```
1. DECLARE
2.    num number;
3.    factorial number;
4.
5. FUNCTION fact(x number)
6. RETURN number
7. IS
8.    f number;
9. BEGIN
10.   IF x=0 THEN
11.      f := 1;
```

```
12.   ELSE
13.      f := x * fact(x-1);
14.   END IF;
15. RETURN f;
16. END;
17.
18. BEGIN
19.   num:= 6;
20.   factorial := fact(num);
21.   dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
22. END;
23. /
```

After the execution of above code at SQL prompt, it produces the following result.

```
Factorial 6 is 720
PL/SQL procedure successfully completed.
```

PL/SQL Drop Function

Syntax for removing your created function:

If you want to remove your created function from the database, you should use the following syntax.

```
1. DROP FUNCTION function_name;
```

PL/SQL Procedure

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

- **Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.
- **Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

How to pass parameters in procedure:

When you want to create a procedure or function, you have to define parameters .There is three

ways to pass parameters in procedure:

1. **IN parameters:** The IN parameter can be referenced by the procedure or function. The value of the parameter cannot be overwritten by the procedure or the function.
2. **OUT parameters:** The OUT parameter cannot be referenced by the procedure or function, but the value of the parameter can be overwritten by the procedure or function.
3. **INOUT parameters:** The INOUT parameter can be referenced by the procedure or function and the value of the parameter can be overwritten by the procedure or function.

5.7 PL/SQL Create Procedure

Syntax for creating procedure:

```
1. CREATE [OR REPLACE] PROCEDURE procedure_name
2.    [ (parameter [,parameter]) ]
3. IS
4.    [declaration_section]
5. BEGIN
6.    executable_section
7. [EXCEPTION
8.    exception_section]
9. END [procedure_name];
```

Create procedure example

In this example, we are going to insert record in user table. So you need to create user table first.

Table creation:

```
1. create table user(id number(10) primary key,name varchar2(100));
```

Now write the procedure code to insert record in user table.

Procedure Code:

```
1. create or replace procedure "INSERTUSER"
2. (id IN NUMBER,
3. name IN VARCHAR2)
4. is
5. begin
```

```
6. insert into user values(id,name);
7. end;
8. /
```

Output:

```
Procedure created.
```

PL/SQL program to call procedure

Let's see the code to call above created procedure.

```
1. BEGIN
2.   insertuser(101,'Rahul');
3.   dbms_output.put_line('record inserted successfully');
4. END;
5. /
```

Now, see the "USER" table, you will see one record is inserted.

ID	Name
101	Rahul

PL/SQL Drop Procedure

Syntax for drop procedure

```
1. DROP PROCEDURE procedure_name;
```

Example of drop procedure

```
1. DROP PROCEDURE pro1;
```

5.8 Package

- A database object that groups related package constructs like Procedures, functions, cursor definitions, variables and constants, exception definitions.
- Package variables and cursors have persistent state.

- Variables retain values and cursors retain contexts and positions for the duration of a user session.
- State persists across a user's calls in one session (not multiple sessions or users).

Parts of a package- Package specification

- Declares (specifies) package constructs, including names and parameters publicly available procedures and functions.

Package body

- May declare additional, private package constructs that are not publicly available. Defines all package constructs (public and private).
- May be replaced without affecting package specification (Breaks dependency chain) Each session has own version of state.

Create package Syntax

When creating packages, the package specification and the package body are created separately. Thus, there are two commands to use: **create package** for the package specification, and **create package body** for the package body.

create [or replace] package packagename
{is | as}
package specification;

A package specification consists of the list of functions, procedures, variables, constants, cursors, and exceptions that will be available to users of the package.

A package body contains the blocks and specifications for all of the public objects listed in the package specification. The package body may include objects that are not listed in the package specification; such objects are said to be private and are not available to users of the package. Private objects may only be called by other objects within the same package body.

The syntax for creating package bodies is

create [or replace] package body packagebody

{is | as}

package body;

The name of the package body should be the same as the name of the package specification.

Example:

```
create or replace package BOOK_MANAGEMENT
as
function OVERDUE_CHARGES(aName IN VARCHAR2) return NUMBER;
procedure NEW_BOOK (aTitle IN VARCHAR2, aPublisher IN VARCHAR2,
aCategoryName IN VARCHAR2);
end BOOK_MANAGEMENT;
create or replace package body BOOK_MANAGEMENT
as
function OVERDUE_CHARGES (aName IN VARCHAR2)
return NUMBER
is
owed_amount NUMBER(10,2);
begin
select SUM(((ReturnedDate-CheckoutDate) -14)*0.20) into owed_amount
from BOOKSHELF_CHECKOUT
where Name = aName and (ReturnedDate-CheckoutDate) > 14;
RETURN(owed_amount);
EXCEPTION
when NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20100,'No books borrowed.');
end OVERDUE_CHARGES;
procedure NEW_BOOK (aTitle IN VARCHAR2,
aPublisher IN VARCHAR2, aCategoryName IN VARCHAR2)
```

```
is
begin
insert into BOOKSHELF (Title, Publisher, CategoryName, Rating)
values (aTitle, aPublisher, aCategoryName, NULL);
delete from BOOK_ORDER where Title = aTitle;
end NEW_BOOK;
end BOOK_MANAGEMENT
```

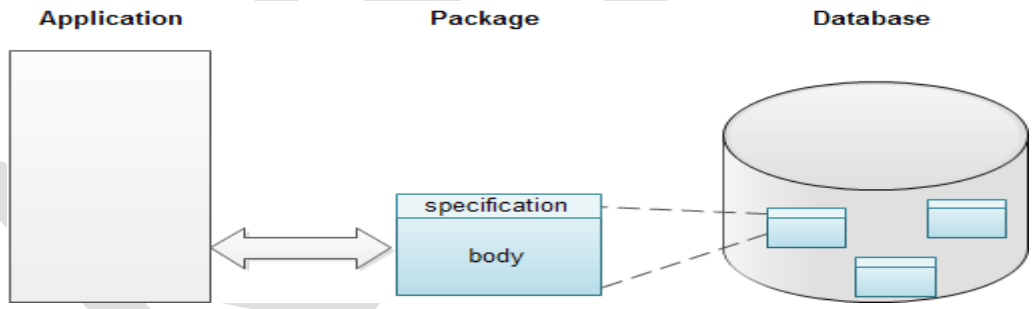
5.8 PL/SQL Package

Introducing to PL/SQL Package

PL/SQL package is a group of related functions, procedures, types, cursors, etc. PL/SQL package is like a library once written stored in the Oracle database and can be used by many applications.

A PL/SQL package has two parts: package specification and package body.

- A package specification is the public interface of your applications. The public means the stored function, procedures, types, etc., are accessible from other applications.
- A package body contains the code that implements the package specification



Replacing Procedures, Functions, and Packages

Procedures, functions, and packages may be replaced via their respective **create or replace** command. Using the **or replace** clause keeps in place any existing grants that have been made for those objects.

Dropping Procedures, Functions, and Packages

- To drop a procedure, use the **drop procedure** command, as follows:

Syntax

drop procedure procedure_name;

Example

drop procedure NEW_BOOK;

- To drop a function, use the **drop function** command, as follows:

Syntax

drop function function_name;

Example

drop function OVERDUE_CHARGES;

- To drop a package (both the specification and the body), use the **drop package** command, as follows:

Syntax

drop package package_name;

Example

drop package BOOK_MANAGEMENT;

- To drop a package body, use the **drop package** command with the **body** clause, as follows:

Syntax

drop package package_body_name;

Example

drop package body BOOK_MANAGEMENT;

Components of Packages

PL/SQL package has two components.

- Package Specification
- Package Body

Package Specification

Package specification consists of a declaration of all the public variables, cursors, objects,

procedures, functions, and exception.

Below are few characteristics of the Package specification.

- The elements which are all declared in the specification can be accessed from outside of the package. Such elements are known as a public element.
- The package specification is a standalone element that means it can exist alone without package body.
- Whenever a package has referred an instance of the package is created for that particular session.
- After the instance is created for a session, all the package elements that are initiated in that instance are valid until the end of the session.

Syntax

```
CREATE [OR REPLACE] PACKAGE <package_name>
IS
<sub_program and public element declaration>
.
.
END <package name>
```

The above syntax shows the creation of package specification.

Package Body

It consists of the definition of all the elements that are present in the package specification. It can also have a definition of elements that are not declared in the specification, these elements are

called private elements and can be called only from inside the package.

Below are characteristics of a package body.

- It should contain definitions for all the subprograms/cursors that have been declared in the specification.
- It can also have more subprograms or other elements that are not declared in specification. These are called private elements.
- It is a dependable object, and it depends on package specification.

- The state of the package body becomes 'Invalid' whenever the specification is compiled. Therefore, it needs to be recompiled each time after the compilation of specification.
- The private elements should be defined first before they are used in the package body.
- The first part of the package is the global declaration part. This includes variables, cursors and private elements (forward declaration) that is visible to the entire package.
- The last part of the package is Package initialization part that executes one time whenever a package is referred first time in the session.

Syntax:

```
CREATE [OR REPLACE] PACKAGE BODY <package_name>
IS
<global_declaration part>
<Private element definition>
<sub_program and public element definition>
.
<Package Initialization>
END <package_name>
```

- The above syntax shows the creation of package body.

Now we are going to see how to refer package elements in the program.

Referring Package Elements

Once the elements are declared and defined in the package, we need to refer the elements to use them.

All the public elements of the package can be referred by calling the package name followed by the element name separated by period i.e. '<package_name>.<element_name>'.

The public variable of the package can also be used in the same way to assign and fetch values from them i.e. '<package_name>.<variable_name>'.

Create Package in PL/SQL

In PL/SQL whenever a package is referred/called in a session a new instance will be created for that package.

Oracle provides a facility to initialize package elements or to perform any activity at the time of this instance creation through 'Package Initialization'.

This is nothing but an execution block that is written in the package body after defining all the package elements. This block will be executed whenever a package is referred for the first time in the session.

Syntax

Syntax:

```
CREATE [OR REPLACE ] PACKAGE BODY <package_name>
IS
<Private element definition>
<sub_program and public element definition>
.
BEGIN
<package_initialization code>;
END <package_name>
```

```
CREATE [OR REPLACE] PACKAGE BODY <package_name>
IS
<Private element definition>
<sub_program and public element definition>
.
BEGINE
<Package Initialization>
END <package_name>
```

Difference between SQL and PL/SQL

SQL(Non-Procedural)	**PL/SQL(Procedural)**
SQL is a single query that is used to perform DML and DDL operations.	PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc.
It is declarative, that defines what needs to be done, rather than how things need to be done.	PL/SQL is procedural that defines how the things needs to be done.
Execute as a single statement.	Execute as a whole block.
Mainly used to manipulate data.	Mainly used to create an application.

Possible Questions

Part-A

(Online – Multiple choice questions)

(Each question carries one mark each)

PART – B (2 Marks)

1. What is PL/SQL?
2. What is Trigger?
3. What is Cursor
4. What is Package?
5. What data types PL/SQL supports?
6. Write the syntax for create package

PART – C (8 Marks)

1. Explain in detail about package specification and package body?
2. What are the main parts to be defined for using a package? Explain them in detail with an example package.
3. What are packages? Explain in detail the components of a PL/SQL package.
4. Write in detail about various data types used in PL/SQL.
5. Describe the characteristics of using functions in PL/SQL. Give the syntax and example for defining a function.
6. Write short notes on the following
   i) Replacing a package ii) Dropping a Package
   iii) Calling subprograms in a package iv) Private and public items in package
7. What are packages? Explain in detail the components of a PL/SQL package.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

Subject: Relational Database Management System Sub.code: 18CTU303

S.No	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____is an oracle object , which holds another object.	procedure	function	package	trigger	**package**
2	subprogram created inside a package is a_____	combined subprogram	collective subprogram	nested subprogram	packaged subprogram	**packaged subprogram**
3	_____is a package that includes a number of procedures and functions that accumulate informationin a buffer so that is can be retrieved later.	output	db_output	DBMS_OUTPUT	DBMS-OUTPUT	**DBMS_OUTPUT**
4	_____put a piece of information in the package buffer followed by an and-of-line marker.	PUT_LINE	DBMS_PUTLINE	DBMS-PUTLINE	PUTLINE	**PUT_LINE**
5	_____is a SQL * PLUS environment parameter that displays the information passed as a parameter to the PUT_LINE function.	SERVEROUTPUT	OUTPUT	SERVER PARAM	BUFFER	**SERVEROUTPUT**
6	_____property enables us to recover any instance of the decomposed relation from corresponding instance of the smaller relations.	dependency preservation	lossless-join	normal forms	decomposition	**lossless-join**

7	PL/SQL stands for _____that is used in Oracle.	Pro Language extension of SQL	Procedural Language extension of SQL	Programming Language extension of SQL	Produce Language extension of SQL	**Procedural Language extension of SQL**
8	A _____ is a database object that groups related package constructs like Procedures, functions, cursor definitions, variables and constants, exception definitions.	Package	Trigger	Cursor	Exception	**Package**
9	When an SQL statement is processed, Oracle creates a memory area known as context area and a _____ is a pointer to this context area.	Trigger	Exception	Cursor	Exception	**Cursor**
10	_____ is invoked by Oracle engine automatically whenever a specified event occurs.	Exception	Cursor	Trigger	Function	**Trigger**
11	An error occurs during the program execution is called _____ in PL/SQL.	Execution	Bug	Exception	Cursor	**Exception**
12	_____marks a sequence of statements that has to be repeated.	procedure	package	subrouting	loop	**loop**
13	The execution of a trigger is also known as _____	Executing the trigger	Firing the trigger	Perform the trigger	implement trigger	**Firing the trigger**
14	The data that is stored in the cursor is called the_____	packages	relations	tables	Active data set	**Active data set**
15	_____is an oracle object, which holds another object.	procedure	function	package	trigger	**package**

16	In a Cursor FOR loop, there is no need for a _____ command	close	exit loop	end loop	end	**close**
17	Which keyword is used to delete the trigger?	delete	drop	remove	truncate	**drop**
18	Which keyword is used to specifies the trigger restriction?	control	restrict	check	when	**when**
19	_____can also be used to keep an audit trail of a table.	trigger	procedure	package	function	**trigger**
20	for each row trigger is known as_____	trigger	tuple trigger	record trigger	row trigger	**row trigger**
21	_____trigger fires after a DML statement is executed.	after	later	next	when	**after**
22	_____allows to access the currently processed row.	another	:new	:fresh	:recent	**:new**
23	_____trigger is fired before execution of a DML statement.	previous	after	before	prior	**before**
24	The module code in the body without its description in the specification is called _____	private module	public module	definition	function	**private module**
25	The objects in the specification section of a package are called_____	private objects	public objects	package variable	modules header	**public objects**
26	The function header comes before the reserved word_____	IN	IS	NOT IN	RETURN	**IS**
27	_____returns a value back to the calling block.	procedure	package	function	trigger	**function**
28	The parameter names in the header of a module are called_____	parameter marker	parameter statements	actual parameters	formal parameters	**formal parameters**

29	The parameter passed in a call statement are called the _____	parameter marker	parameter statements	actual parameters	formal parameters	**actual parameters**
30	A _____is the building block of modular programming.	packages	procedure	cursor	exception	**procedure**
31	_____is a logically grouped set of SQL and PL/SQL statements that perofrm a specific task.	procedure	cursors	package	trigger	**procedure**
32	The _____used to roll the transaction back to a certain point without rolling back the entire transaction.	commit	rollback	savepoint	revoke	**savepoint**
33	_____deals with handling of errors that arise during execution of the data manipulation statements which make up the PL/SQL code block.	begin	exception	end	declare	**exception**
34	User defined cursor is known as_____	implicit cursor	explicit cursor	internal cursor	external cursor	**explicit cursor**
35	If the Oracle engine opened a cursor for its internal processing it is known as_____	explicit cursor	implicit cursor	internal cursor	external cursor	**implicit cursor**
36	When creating packages, the package _____ and the package body are created separately.	declaration	initialization	specification	functions	**specification**

37	A package body contains the _____ and specifications for all of the public objects listed in the package specification	block	package	tigger	procedure	**block**
38	The loop statement must end with an _____statement	end loop	loop	end	goto	**end loop**
39	The loop statement must end with an _____statement	end loop	loop	end	goto	**end loop**
40	Which keyword should be placed before the first statement in the sequence of PL/SQL block?	loop	begin	declare	for loop	**begin**
41	Which attribute is used to count the number of rows affected by an insert, update, delete statement?	%COUNT	%ROWCOUNT	%CALCULATE	%ROWCALCULATE	**%ROWCOUNT**
42	Which attribute is used to open and close the cursor automatically?	%FOUND	%NOTFOUND	%ISOPEN	%ROWCOUNT	**%ISOPEN**
43	An _____ cursor is automatically declared by Oracle every time an SQL statement is executed.	implicit cursor	explicit cursor	internal cursor	external cursor	**implicit cursor**
44	A cursor is defined in _____part of a PL/SQL block.	begin	declarative part	end	exception	**declarative part**
45	_____are composite types that have internal components that can be manipulated individually, such as the elements of an array, record, or table.	collections and records	composite attribute and single attribute	collection types	nested tables	**collections and records**

46	_____statement disables the cursor	disable cursor	close cursor	end cursor	remove cursor	**close cursor**
47	_____are similar to C or Pascal pointers, which hold the memory location (address) of some item instead of the item itself.	trigger	procedure	pointers	cursor variables	**cursor variables**
48	_____can be a procedure or function	stored procedure	query	nested program	subprogram	**subprogram**
49	subprogram created inside a PL/SQL block is a_____	combined subprogram	collective subprogram	nested subprogram	packaged subprogram	**nested subprogram**
50	_____allows programmers to issue user-defined error messages.	raise_error	application error	raise_application_error	error_message	**raise_application_error**
51	What are the ranges for error numbers in the trigger?	2000 to 2099	20001-29000	-20000 to -20999	5000-10000	**-20000 to -20999**
52	Procedure and functions are stored in the _____	oracle database	DBMS	repository	data warehouse	**oracle database**
53	When the procedure or function is invoked, the oracle engine loads the compiled procedure or function in a memory are called_____	process global area	procedure global area	System global area	function global area	**System global area**
54	How many copies are needed to be loaded for execution by multiple user?	1	2	3	4	**1**
55	Orcale provides some attributes known as Implicit cursor's attributes to check the status of DML operations and the following one is not	%ISOPEN	%FOUND	%ROWCOUNT	%COUNT	**%COUNT**

56	When a subprogram calls itself, it is called recursive call and the process is known as	procedure	function	recursion	called function	**recursion**
57	_____is a superset of SQL.	PSQL	PL/SQL	PRO/SQL	RA	**PL/SQL**
58	code block starts with _____section in PL/SQL	begin	end	start	declare	**declare**
59	The _____section marks the end of a PL/SQL block.	Terminate	end	last	next	**end**
60	_____ is used to search for values that are within a set of values.	AND	EXISTS	BETWEEN	ANY	**BETWEEN**
61	It returns the number of rows affected by DML statements like INSERT, DELETE, and UPDATE or returned by a SELECT INTO statement.	%ISOPEN	%ROWCOUNT	%FOUND	%NOTFOUND	**%ROWCOUNT**

Reg. No ----------

[18CTU303]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

FIRST INTERNAL EXAMINATION, JULY 2019

Third Semester

COMPUTER TECHNOLOGY

RELATIONAL DATABASE MANAGEMENT SYSTEM

Class: II B.Sc. CT **Time: 2 Hours**

Date/Session: /7/19 N **Maximum: 50**

PART-A [20 * 1 = 20 Marks]

(Answer all of the following)

1. The data stored in database at a particular moment of time is called _____ of database.
   a) schema b) instance c) model d) system
2. The overall design of a database is called
   a) Model b) Instance
   c) Design d) Schema
3. What is a SQL?
   a) Segment Query Language b) Select Query Language
   c) Structured Query Language d) Secured Query Language
4. _____ is used to define that how the data will be stored in a block.
   a) View Schema b) Conceptual Schema c) Physical Schema d) External Schema
5. Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level is called as _____
   a) Data Independence b) Data Security c) Data Integrity d) Data Consistency
6. The Grant task is in _____.
   a) DDL b) DML c) DCL d) TCL
7. The task used to remove all records from a table
   a) truncate b) create c) alter d) drop
8. _____ is used to define database structure or pattern.
   a) TCL b) DML c) DDL d) DCL
9. In E-R Model entities are represented by _____
   a) Rectangle b) Rhombus c) Ellipse d) Square
10. The database model organises data into a tree-like-structure, with a single root, to which all the other data is linked is
    a) Network Model b) Relational Model c) E-R Model d) Hierarchical Model
11. DBMS states are _____
    a) Online, Offline b) Submit, Run c) Ready, Run d) Submit, block.
12. _____ are the one who really use and take the benefits of database.
    a) Database administrators b) Database users
    c) Database assistants d) Database associates
13. DCL stands for _____
    a) Data Consistency Language b) Data Commit Language

c) Data Concurrency Language d) Data Control Language

14. The task come under DML is _____
    a) Select b) Commit c) Grant d) Create
15. The term DBA is _____
    a) Database Assistant b) Database Administrator
    c) Departmental Based Assistant d) Data Based Administration
16. In DBMS, which one contains metadata i.e., data about the database?
    a) Database b) RDBMS c) Data Dictionary d) DBA
17. _____ is the most widely used data model.
    a) Network Model b) E-R Model c) Hierarchical Model d) Relational Model
18. _____ is a visual representation of data that describes how data is related to each other.
    a) Entity Sets b) Key Attribute c) Binary Relationship d) E-R Diagram
19. The tables are also called as _____
    a) Relations b) Entities
    c) E-R Mosel d) Network Model
20. _____ is one of the candidate keys chosen by the database designer to uniquely identify the entity set.
    a) Super Key b) Primary Key
    c) Foreign Key d) Composite Key

PART-B **[3 * 2 = 6 Marks]**

(Answer all of the following)

21. What is DBMS?
22. What is DML?
23. What is Entity?

PART-C **[3* 8 = 24 Marks]**

(Answer ALL of the following)

23. a. Explain about Characteristics of DBMS.

(OR)

b. Describe about DBMS Languages.

25. a. Explain about (i) Data Independence (ii) Data Dictionary.

(OR)

b. Discuss about (i) Database Users (ii) Database Administrators.

26. a. Explain in detail about Data Models.

(OR)

b. Explain E - R Diagram.

Reg.No ----------

[18CTU303]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

FIRST INTERNAL EXAMINATION, JULY 2019

Third Semester

COMPUTER TECHNOLOGY

RELATIONAL DATABASE MANAGEMENT SYSTEM

Answer Key

Class: II B.Sc. CT **Time : 2 Hours**

Date/Session: /7/19 N **Maximum : 50**

PART-A **[20 * 1 = 20 Marks]**

(Answer all of the following)

RELATIONAL DATABASE MANAGEMENT SYSTEM [18CTU303]

CIA – I KEYS

1. b) Instance
2. d) Schema
3. c) Structured Query Language
4. c) Physical Schema
5. a) Data Independence
6. c) DCL
7. a) truncate
8. c) DDL
9. a) Rectangle
10. d) Hierarchical Model
11. a) Online, Offline
12. b) Database users
13. d) Data Control Language
14. a) Select

15. b) Database Administrator
16. c) Data Dictionary
17. d) Relational Model
18. d) E-R Diagram
19. a) Relations
20. b) Primary Key

PART-B [3 * 2 = 6Marks]

(Answer all of the following)

21. What is DBMS?

A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate this data. A DBMS relieves users of framing programs for data maintenance.

22. What is DML?
A data manipulation language (DML) is a family of computer languages including commands permitting users to manipulate data in a database. This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data. DML is mostly incorporated in SQL databases.
23. What is Entity?
An **entity** is any object in the system that we want to model and store information about. **Entities** are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the **database**. Some specific examples of **entities** are Employee, Student, Lecturer.

PART-C [3* 8 = 24Marks]

(Answer ALL of the following)

24. a. Explain about Characteristics of DBMS.

Stores any kind of data

A database management system should be able to store any kind of data. It should not be restricted to the employee name, salary and address. Any kind of data that exists in the real world can be stored in DBMS because we need to work with all kinds of data that is present around us.

Support ACID Properties

Any DBMS is able to support ACID (Accuracy, Completeness, Isolation, and Durability) properties. It is made sure is every DBMS that the real purpose of data should not be lost while performing transactions like delete, insert an update. Let us take an example; if an

employee name is updated then it should make sure that there is no duplicate data and no mismatch of student information.

Represents complex relationship between data

Data stored in a database is connected with each other and a relationship is made in between data. DBMS should be able to represent the complex relationship between data to make the efficient and accurate use of data.

Backup and recovery

There are many chances of failure of whole database. At that time no one will be able to get the database back and for sure company will be in a big loss. The only solution is to take backup of database and whenever it is needed, it can be stored back. All the databases must have this characteristic.

Structures and described data

A database should not contains only the data but also all the structures and definitions of the data. This data represent itself that what actions should be taken on it. These descriptions include the structure, types and format of data and relationship between them.

Data integrity

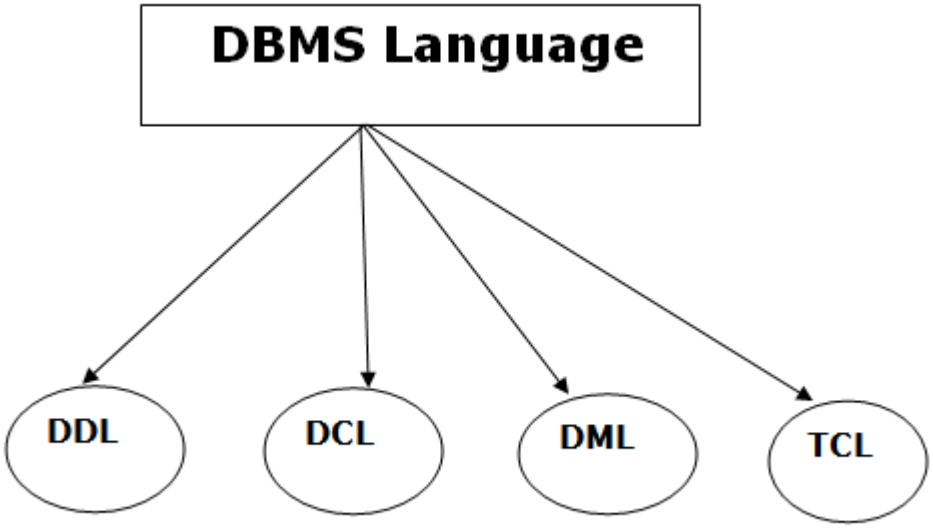
This is one of the most important characteristics of database management system. Integrity ensures the quality and reliability of database system. It protects the unauthorized access of database and makes it more secure. It brings only the consistence and accurate data into the database.

Concurrent use of database

There are many chances that many users will be accessing the data at the same time. They may require altering the database system concurrently. At that time, DBMS supports them to concurrently use the database without any problem.

(OR)

b. Describe about DBMS Languages.



1. Data Definition Language

- **DDL** stands for **D**ata **D**efinition **L**anguage. 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 **D**ata **M**anipulation **L**anguage. 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 **D**ata **C**ontrol **L**anguage. 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:

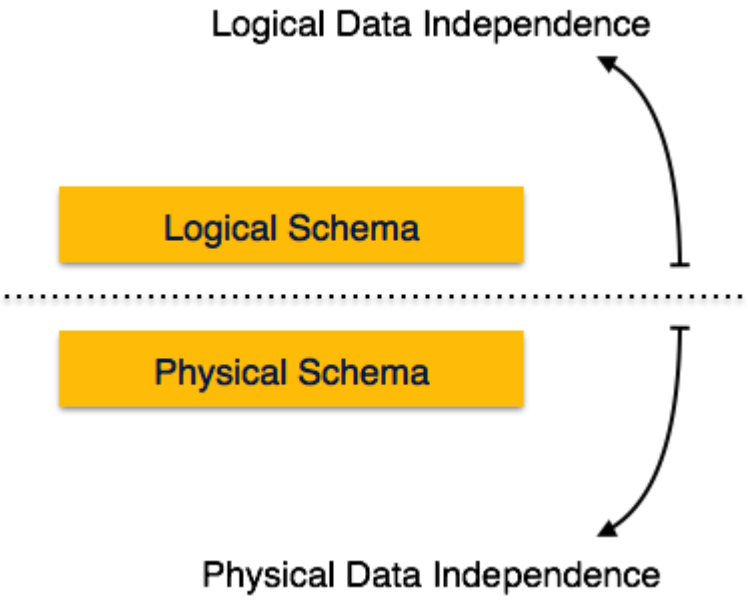
- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

25. a. Explain about (i) Data Independence (ii) Data Dictionary.

If a database system is not multi-layered, then it becomes difficult to make any changes in the database system. Database systems are designed in multi-layers as we learnt earlier.

Data Independence

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.



Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

Data Dictionary consists of database metadata. It has records about objects in the database.

What Data Dictionary consists of

Data Dictionary consists of the following information:

1. Name of the tables in the database
2. Constraints of a table i.e. keys, relationships, etc.
3. Columns of the tables that related to each other
4. Owner of the table
5. Last accessed information of the object
6. Last updated information of the object

An example of Data Dictionary can be personal details of a student:

Example

<StudentPersonalDetails>

Student_ID	**Student_Name**	**Student_Address**	**Student_City**

The following is the data dictionary for the above fields:

Types of Data Dictionary

Here are the two types of data dictionary:

Active Data Dictionary

The DBMS software manages the active data dictionary automatically. The modification is an automatic task and most RDBMS has active data dictionary. It is also known as integrated data dictionary.

Passive Data Dictionary

Managed by the users and is modified manually when the database structure change. Also known as non-integrated data dictionary.

(OR)

b. Discuss about (i) Database Users (ii) Database Administrators.

Different Types of Database Users in DBMS

Application Programmers

As its name shows, application programmers are the one who writes application programs that uses the database. These application programs are written in programming languages like COBOL or PL (Programming Language 1), Java and fourth generation language. These programs meet the user requirement and made according to user requirements. Retrieving information, creating new information and changing existing information is done by these application programs.

Also See: Advantages of Database Management System

They interact with DBMS through DML (Data manipulation language) calls. And all these functions are performed by generating a request to the DBMS. If application programmers are not there then there will be no creativity in the whole team of Database.

End Users

End users are those who access the database from the terminal end. They use the developed applications and they don't have any knowledge about the design and working of database. These are the second class of users and their main motto is just to get their task done. There are basically two types of end users that are discussed below.

- **Casual User**

These users have great knowledge of query language. Casual users access data by entering different queries from the terminal end. They do not write programs but they can interact with the system by writing queries.

- **Naive**

Any user who does not have any knowledge about database can be in this category. There task is to just use the developed application and get the desired results. For example: Clerical staff in any bank is a naïve user. They don't have any dbms knowledge but they still use the database and perform their given task.

Also See: What is traditional File Processing System.

DBA (Database Administrator)

DBA can be a single person or it can be a group of person. Database Administrator is responsible for everything that is related to database. He makes the policies, strategies and provides technical supports.

System Analyst

System analyst is responsible for the design, structure and properties of database. All the requirements of the end users are handled by system analyst. Feasibility, economic and technical aspects of DBMS is the main concern of system analyst.

A ***d**ata**b**ase **a**dministrator* (***DBA***) is a specialized computer systems administrator who maintains a successful database environment by directing or performing all related activities to keep the data secure. The top responsibility of a DBA professional is to maintain data integrity. This means the DBA will ensure that data is secure from unauthorized access but is available to users.

A database administrator will often have a working knowledge and experience with a wide range of database management products such as Oracle-based software, SAP and SQL, in addition to having obtained a degree in Computer Science and practical field experience and additional, related IT certifications.

Computer systems design and related services firms, banks, insurance companies, universities and health care are a few of the many different types of industries that a database administrator will find work.

In addition to being responsible for backing up systems in case of power outages or other disasters, a DBA is also frequently involved in tasks related to training employees in database management and use, designing, implementing, and maintaining the database system and establishing policies and procedures related to the organization's data management policy.

25. a. Explain in detail about Data Models.

DBMS Database Models

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

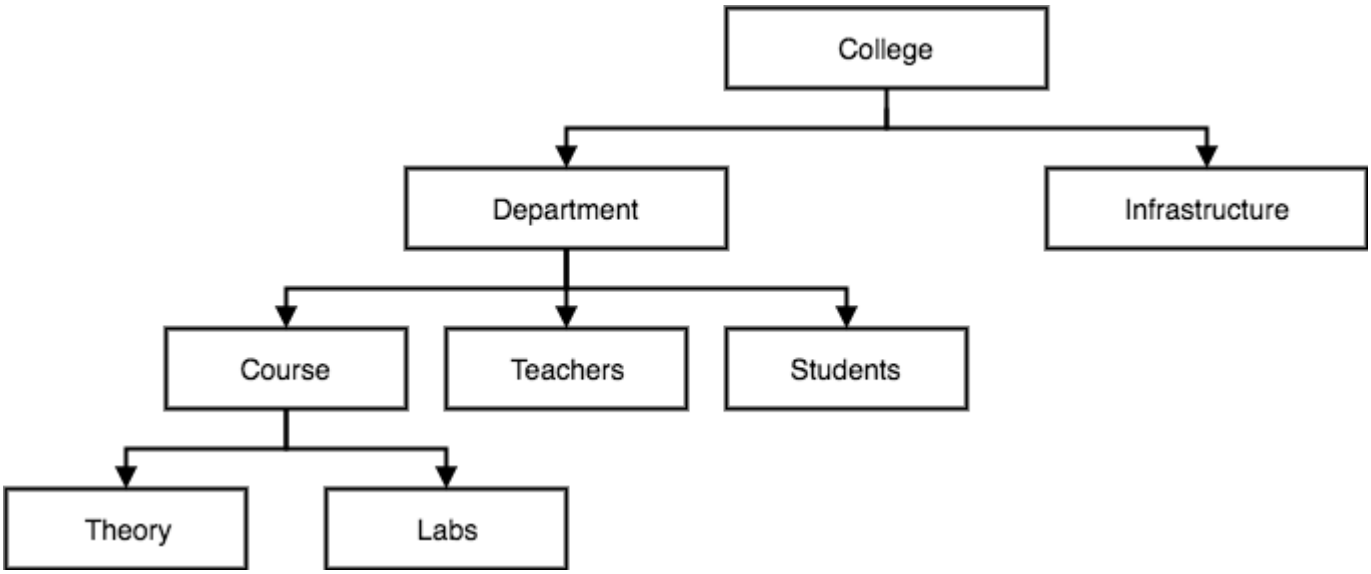
Hierarchical Model

This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The heirarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

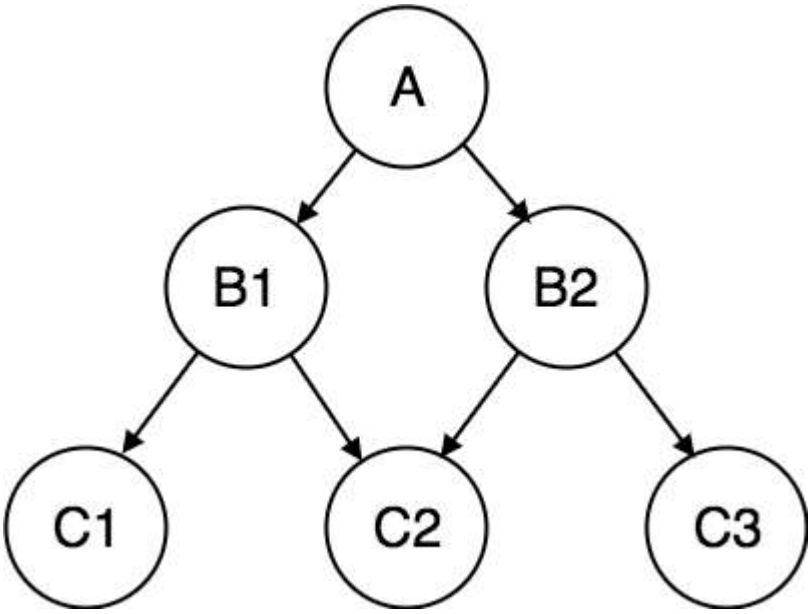


Network Model

This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.



Entity-relationship Model

In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

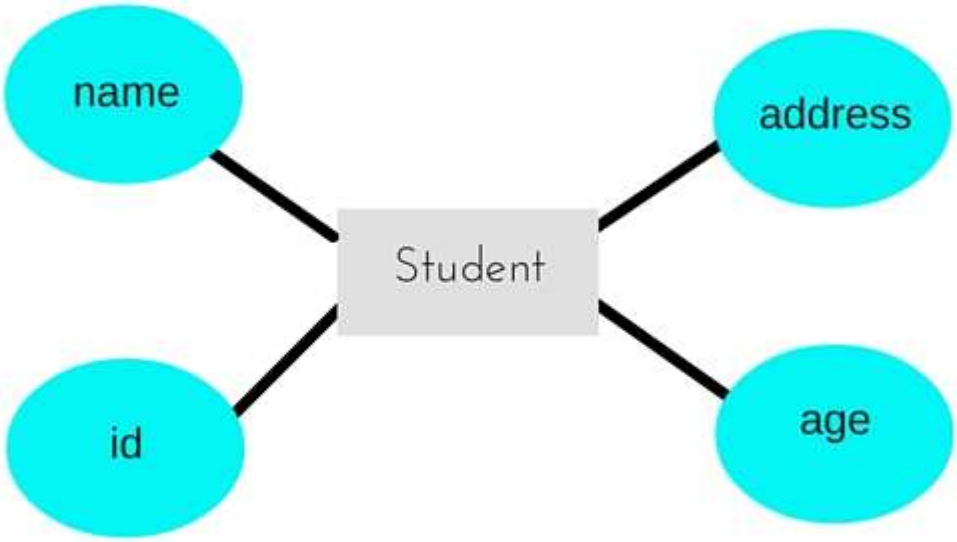
Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

This model is good to design a database, which can then be turned into tables in relational model(explained below).

Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.

Relationships can also be of different types. To learn about E-R Diagrams in details, click on the link.



Relational Model

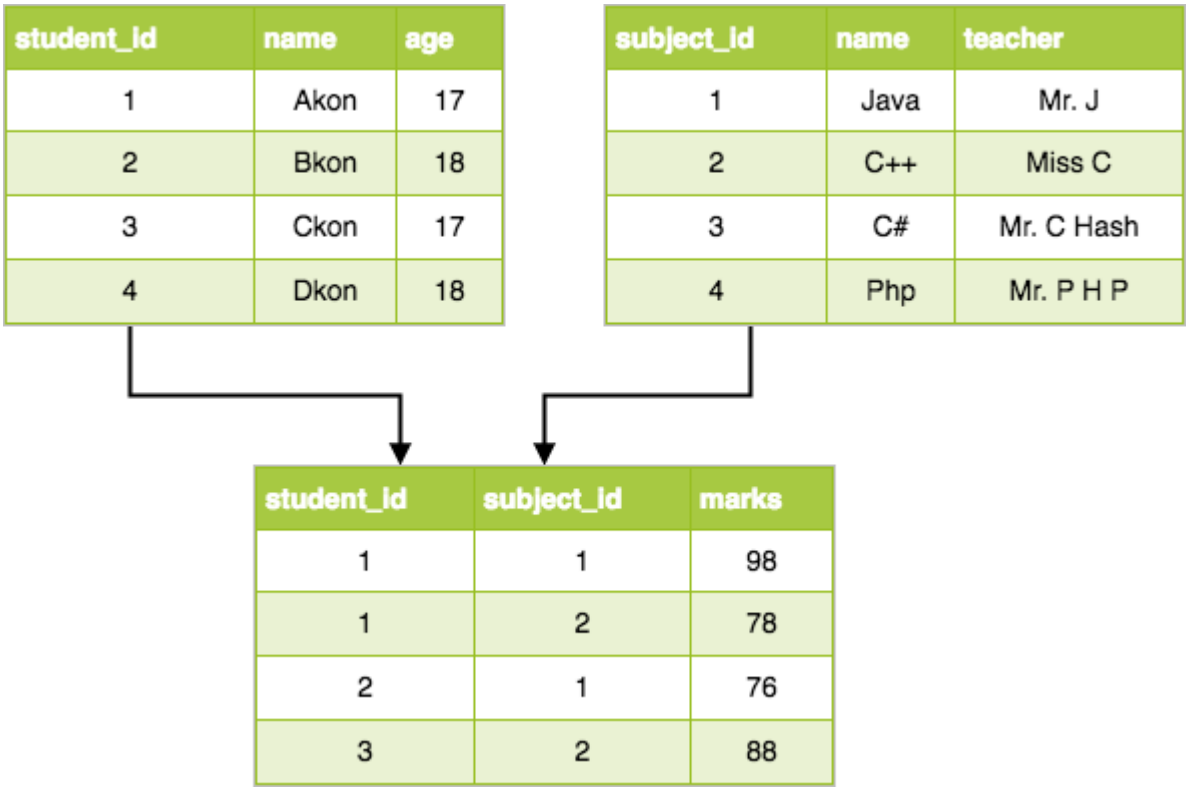
In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.

Hence, tables are also known as **relations** in relational model.

In the coming tutorials we will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

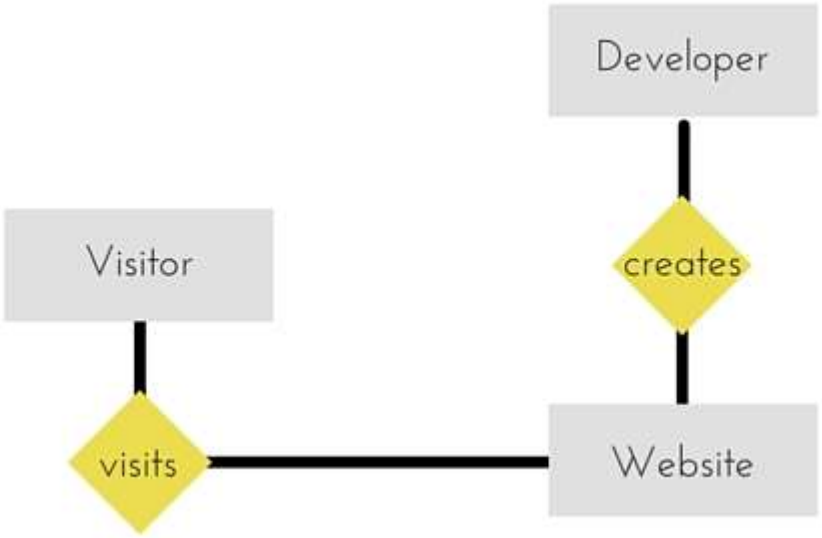
(OR)

b. Explain E - R Diagram.

Working with ER Diagrams

ER Diagram is a visual representation of data that describes how data is related to each other. In ER Model, we disintegrate data into entities, attributes and setup relationships between entities, all this can be represented visually using the ER diagram.

For example, in the below diagram, anyone can see and understand what the diagram wants to convey: *Developer develops a website, whereas a Visitor visits a website*.



Components of ER Diagram

Entitiy, Attributes, Relationships etc form the components of ER Diagram and there are defined symbols and shapes to represent each one of them.

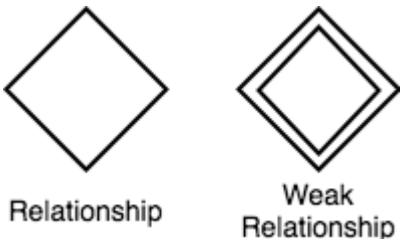
Let's see how we can represent these in our ER Diagram.

Entity

Simple rectangular box represents an Entity.

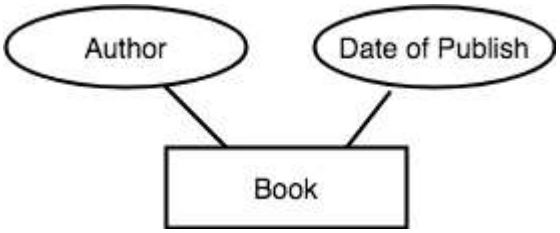
Relationships between Entities - Weak and Strong

Rhombus is used to setup relationships between two or more entities.



Attributes for any Entity

Ellipse is used to represent attributes of any entity. It is connected to the entity.



Weak Entity

A weak Entity is represented using double rectangular boxes. It is generally connected to another entity.

Key Attribute for any Entity

To represent a Key attribute, the attribute name inside the Ellipse is underlined.

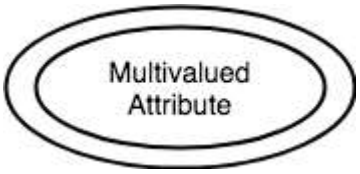
Derived Attribute for any Entity

Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth.

To represent a derived attribute, another dotted ellipse is created inside the main ellipse.

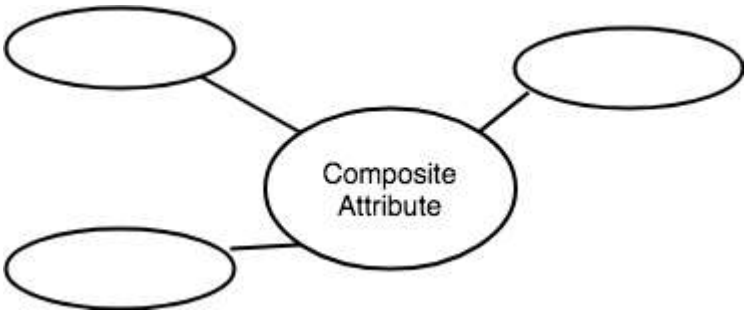
Multivalued Attribute for any Entity

Double Ellipse, one inside another, represents the attribute which can have multiple values.



Composite Attribute for any Entity

A composite attribute is the attribute, which also has attributes.



ER Diagram: Entity

An **Entity** can be any object, place, person or class. In ER Diagram, an **entity** is represented using rectangles. Consider an example of an Organisation- Employee, Manager, Department, Product and many more can be taken as entities in an Organisation.

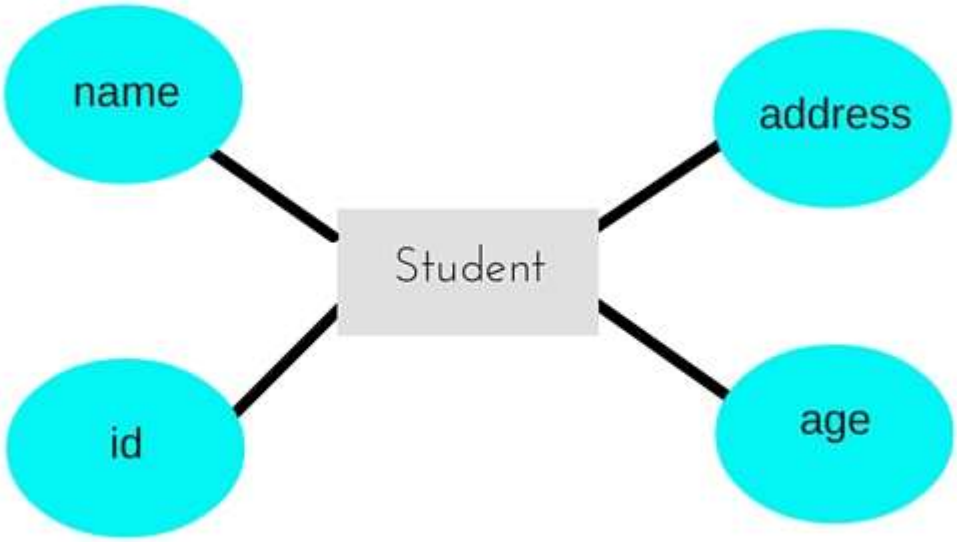
The yellow rhombus in between represents a relationship.

ER Diagram: Weak Entity

Weak entity is an entity that depends on another entity. Weak entity doesn't have anay key attribute of its own. Double rectangle is used to represent a weak entity.

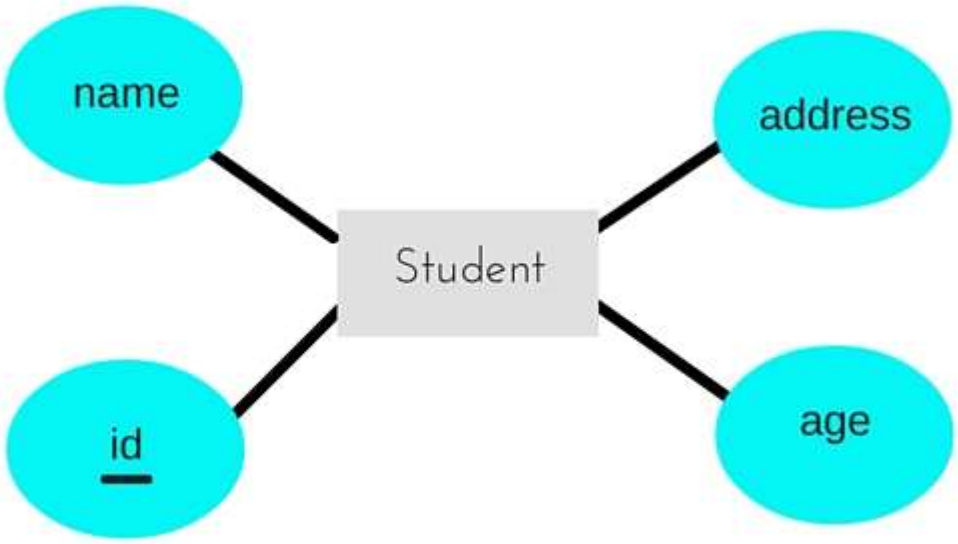
ER Diagram: Attribute

An **Attribute** describes a property or characterstic of an entity. For example, **Name**, **Age**, **Address** etc can be attributes of a **Student**. An attribute is represented using eclipse.



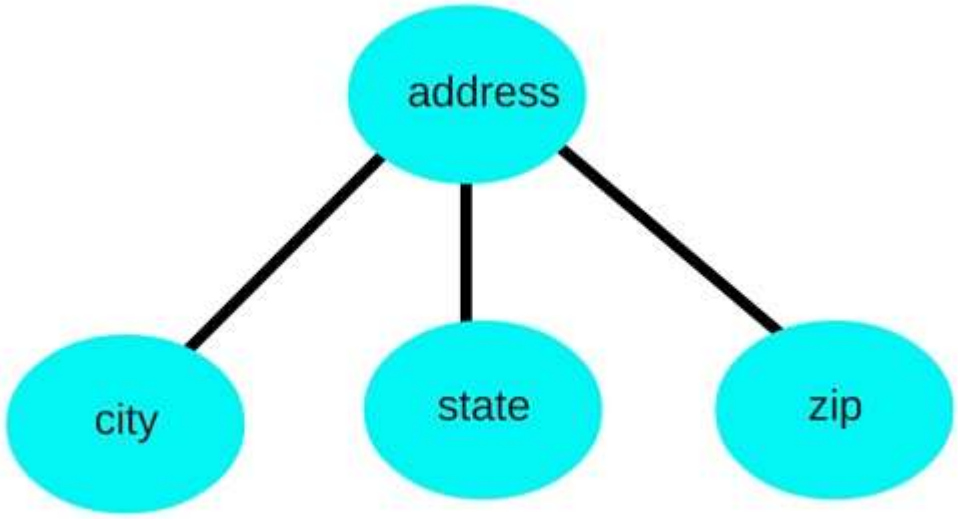
ER Diagram: Key Attribute

Key attribute represents the main characterstic of an Entity. It is used to represent a Primary key. Ellipse with the text underlined, represents Key Attribute.



ER Diagram: Composite Attribute

An attribute can also have their own attributes. These attributes are known as **Composite** attributes.



ER Diagram: Relationship

A Relationship describes relation between **entities**. Relationship is represented using diamonds or rhombus.

There are three types of relationship that exist between Entities.

1. Binary Relationship
2. Recursive Relationship
3. Ternary Relationship

ER Diagram: Binary Relationship

Binary Relationship means relation between two Entities. This is further divided into three types.

One to One Relationship

This type of relationship is rarely seen in real world.

The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in real-world relationships.

One to Many Relationship

The below example showcases this relationship, which means that 1 student can opt for many courses, but a course can only have 1 student. Sounds weird! This is how it is.

Many to One Relationship

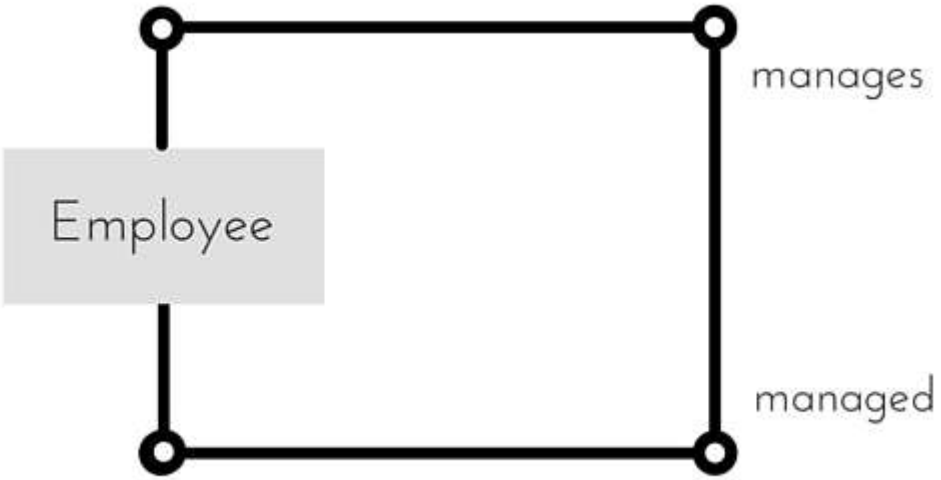
It reflects business rule that many entities can be associated with just one entity. For example, Student enrolls for only one Course but a Course can have many Students.

Many to Many Relationship

The above diagram represents that one student can enroll for more than one courses. And a course can have more than 1 student enrolled in it.

ER Diagram: Recursive Relationship

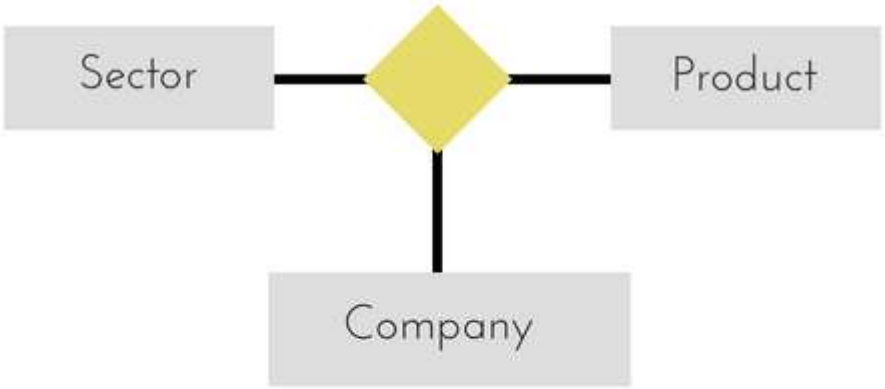
When an Entity is related with itself it is known as **Recursive** Relationship.



ER Diagram: Ternary Relationship

Relationship of degree three is called Ternary relationship.

A Ternary relationship involves three entities. In such relationships we always consider two entites together and then look upon the third.



- The above relationship involves 3 entities.
- Company operates in Sector, producing some Products.

For example, in the diagram above, we have three related entities, **Company**, **Product**and **Sector**. To understand the relationship better or to define rules around the model, we should relate two entities and then derive the third one.

A **Company** produces many **Products**/ each product is produced by exactly one company.

A **Company** operates in only one **Sector** / each sector has many companies operating in it.

Considering the above two rules or relationships, we see that although the complete relationship involves three entities, but we are looking at two entities at a time.

Reg.No ----------

[18CTU303]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021

SECOND INTERNAL EXAMINATION, AUGUST 2019

Third Semester

COMPUTER TECHNOLOGY

RELATIONAL DATABASE MANAGEMENT SYSTEMS

Class: II B.Sc. CT **Time : 2**

HoursDate/Session: .8.19/ N **Maximum : 50 Marks**

PART-A [20 * 1 = 20 Marks]

(Answer ALL of the following.)

1. ____ attributes cannot be further divided into smaller components
   a) Composite b) Simple c) Derived d) Stored

2. Derived attributes are depicted in the E-R diagram with
   a) Double-lined ellipse b) Single-lined ellipse c) Dotted line d) Dashed ellipse

3. _____clause is used to modify a particular row
   a) Update b) Insert c) Delete d) Modify

4. Which one is not the component of DBMS?
   a) Query Processor b) Data Manager c) Database Engine d) Data User

5. A _____ is a set of permitted values for an attribute in table.
   a) Instance b) Domain c) Schema d) Key

6. SQL Non Equi join does not use the symbol
   a) > b) != c) = d) >=

7. Which key is used to implement referential Integrity?
   a) Primary key b) Candidate key c) Foreign key d) Super key

8. _____is a collection of high-level data description constructs that hide many low-level storage details
   a) Network Model b) Relational Model c) E-R Model d) Hierarchical Model

9. Lossless decomposition is_____
   a) Reversible b) Irreversible c) Either a) or b) d) Not applicable

10. Project operation results in projection of _____
    a) Tuples b) Columns c) Relations d) None of the above

11. A table can have only one _____
    a) Candidate key b) Primary Key c) Alternate key d) Foreign key

12. What is the expansion of BCNF?
    a) Boyd – Crowell Normal Form b) Boyce – Codd Normal Form
    c) Boyd – Codd Normal Form d) Bayes - Codd Normal Form

13. Which of the following is the process of increasing redundancy in the database either for convenience or to improve performance?
    a) Normalization b) Optimization c) Demoralization d) Decomposition

14. Which of the following is relational algebra operation?
    a) INTERSECTION b) SELECT c) CARTESIAN PRODUCT d) UNION

15. Which one of the following is not outer join?
    a) Full Outer Join b) Left Outer Join c) Right Outer Join d) Non Equi Join

16. In self join how many tables are involved?
    a) 1 b) 2 c) 3 d) 4

17. In which normal form transitive dependency is eliminated?
    a) 1NF b) 2NF c) 3NF d) 4NF

18. _____ is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key.
    a) 2NF b) 3NF c) 4NF d) 5NF

19. If two relations have 5 and 10 tuples respectively, then what will be the number in the CARTESIAN PRODUCT?
    a) 5 b) 10 c) 15 d) 50

20. Which of the following aggregate function does not ignore nulls its results?
    a) COUNT b) COUNT (*) c) MAX d) MIN

PART-B **[3 * 2 = 6 Marks]**

(Answer all of the following)

21. What is lossless decomposition? Give example

22. What are SQL non equi joins?

23. What is Theta (θ) join?

PART-C **[3* 8 = 24 Marks]**

(Answer ALL of the following.)

24 a. Explain RDBMS Concepts.

(OR)

b. Explain different relational algebra operation with example

25. a. Explain the components of DBMS.

(OR)

b. Discuss different forms of SELECT with examples.

26. a. Explain 1 NF and 2 NF with a suitable examples.

(OR)

b. What is join operation? Explain different join operation with suitable table.

Reg.No ----------

[18CTU303]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021

SECOND INTERNAL EXAMINATION, AUGUST 2019

Third Semester

COMPUTER TECHNOLOGY

RELATIONAL DATABASE MANAGEMENT SYSTEMS

Class: II B.Sc. CT **Time : 2 HoursDate/Session: .8.19/ N** **Maximum : 50 Marks**

PART-A [20 * 1 = 20 Marks]

(Answer ALL of the following.)

1. ____ attributes cannot be further divided into smaller components
   a) Composite **b) Simple** c) Derived d) Stored

2. Derived attributes are depicted in the E-R diagram with
   a) Double-lined ellipse b) Single-lined ellipse **c) Dotted line** d) Dashed ellipse

3. _____clause is used to modify a particular row
   a) Update b) Insert c) Delete d) Modify

4. Which one is not the component of DBMS?
   a) Query Processor b) Data Manager c) Database Engine **d) Data User**

5. A _____ is a set of permitted values for an attribute in table.
   a) Instance **b) Domain** c) Schema d) Key

6. SQL Non Equi join does not use the symbol
   a) > b) != **c) =** d) >=

7. Which key is used to implement referential Integrity?
   a) Primary key b) Candidate key **c) Foreign key** d) Super key

8. _____is a collection of high-level data description constructs that hide many low-level storage details
   a) Network Model b) Relational Model c) E-R Model d) Hierarchical Model

9. Lossless decomposition is_____
   a) Reversible b) Irreversible c) Either a) or b) d) Not applicable

10. Project operation results in projection of _____
    a) Tuples **b) Columns** c) Relations d) None of the above

11. A table can have only one _____
    a) Candidate key **b) Primary Key** c) Alternate key d) Foreign key

12. What is the expansion of BCNF?
a) Boyd – Crowell Normal Form **b) Boyce – Codd Normal Form**
c) Boyd – Codd Normal Form d) Bayes - Codd Normal Form

13. Which of the following is the process of increasing redundancy in the database either for convenience or to improve performance?
a) Normalization b) Optimization **c) Demoralization** d) Decomposition

14. Which of the following is relational algebra operation?
a) INTERSECTION b) SELECT c) CARTESIAN PRODUCT d) UNION

15. Which one of the following is not outer join?
a) Full Outer Join b) Left Outer Join c) Right Outer Join **d) Non Equi Join**

16. In self join how many tables are involved?
a) 1 b) 2 c) 3 d) 4

17. In which normal form transitive dependency is eliminated?
a) 1NF b) 2NF **c) 3NF** d) 4NF

18. _____ is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key.
a) 2NF b) 3NF **c) 4NF** d) 5NF

19. If two relations have 5 and 10 tuples respectively, then what will be the number in the CARTESIAN PRODUCT?
a) 5 b) 10 c) 15 **d) 50**

20. Which of the following aggregate function does not ignore nulls its results?
a) COUNT b) COUNT (*) c) MAX d) MIN

PART-B **[3 * 2 = 6 Marks]**

(Answer all of the following)

21. What is lossless decomposition? Give example

22. What are SQL non equi joins?

23. What is Theta (θ) join?

PART-C **[3* 8 = 24 Marks]**

(Answer ALL of the following.)

24 a. Explain RDBMS Concepts.

RDBMS stands for relational database management system. A relational model can be represented as a table of rows and columns. A relational database has following major components:

1. Table
2. Record or Tuple
3. Field or Column name or Attribute
4. Domain
5. Instance

6. Schema
7. Keys

1. Table

A table is a collection of data represented in rows and columns. Each table has a name in database. For example, the following table "STUDENT" stores the information of students in database.

Table: STUDENT

Student_Id	Student_Name	Student_Addr	Student_Age
101	Chaitanya	Dayal Bagh, Agra	27
102	Ajeet	Delhi	26
103	Rahul	Gurgaon	24
104	Shubham	Chennai	25

2. Record or Tuple

Each row of a table is known as record. It is also known as tuple. For example, the following row is a record that we have taken from the above table.

102	Ajeet	Delhi	26

3. Field or Column name or Attribute

The above table "STUDENT" has four fields (or attributes): Student_Id, Student_Name, Student_Addr & Student_Age.

4. Domain

A domain is a set of permitted values for an attribute in table. For example, a domain of month-of-year can accept January, February,…December as values, a domain of dates can accept all possible valid dates etc. We specify domain of attribute while creating a table.

An attribute cannot accept values that are outside of their domains. For example, in the above table "STUDENT", the Student_Id field has integer domain so that field cannot accept values that are not integers for example, Student_Id cannot has values like, "First", 10.11 etc.

5. Instance and Schema

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

Definition of 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.

6. Keys

Key plays an important role in relational database; it is used for identifying unique rows from table. It also establishes relationship among tables.

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 combination of columns that uniquely identifies any row within a table.

Candidate Key – A candidate key is a closely related concept where the super key is reduced to the minimum number of columns required to uniquely identify each row.

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.

(OR)

b. Explain different relational algebra operation with example

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

We will discuss all these operations in the following sections.

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation − $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.

For example −

$\sigma_{subject = "database"}(Books)$

Output − Selects tuples from books where subject is 'database'.

$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$

Output − Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$

Output − Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation − $\prod_{A1, A2, An}(r)$

Where A_1, A_2 , A_n are attribute names of relation **r**.

Duplicate rows are automatically eliminated, as relation is a set.

For example −

$\prod_{subject, author}(Books)$

Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪)

It performs binary union between two given relations and is defined as −

$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$

Notation − r U s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$\prod_{author}(Books) \cup \prod_{author}(Articles)$

Output − Projects the names of the authors who have either written a book or an article or both.

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation − **r** − **s**

Finds all the tuples that are present in **r** but not in **s**.

$\prod_{author}(Books) - \prod_{author}(Articles)$

Output − Provides the name of authors who have written books but not articles.

Cartesian Product (X)

Combines information of two different relations into one.

Notation − r X s

Where **r** and **s** are relations and their output will be defined as −

r X s = { q t | q ∈ r and t ∈ s}

$\sigma_{author = 'balamurugan'}(Books\ X\ Articles)$

Output − Yields a relation, which shows all the books and articles written by balamurugan.

Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

```
SELECT column-name-list
```

```
FROM
table-name1 CROSS JOIN table-name2;
```

Example of Cross JOIN

Following is the **class** table,

ID	NAME
1	abhi
2	adam
4	alex

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

`Cross` JOIN query will be,

```
SELECT * FROM
class CROSS JOIN class_info;
```

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	1	DELHI
4	alex	1	DELHI
1	abhi	2	MUMBAI
2	adam	2	MUMBAI
4	alex	2	MUMBAI
1	abhi	3	CHENNAI
2	adam	3	CHENNAI
4	alex	3	CHENNAI

As you can see, this join returns the cross product of all the records present in both the tables.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** ρ.

Notation − ρ_x (E)

Where the result of expression **E** is saved with name of **x**.

Additional operations are −

- Set intersection
- Assignment
- Natural join

24. a. Explain the components of DBMS.

DBMS have several components, each performing very significant tasks in the database management system environment. Below is a list of components within the database and its environment.

Software

This is the set of programs used to control and manage the overall database. This includes the DBMS software itself, the Operating System, the network software being used to share the data among users, and the application programs used to access data in the DBMS.

Hardware

Consists of a set of physical electronic devices such as computers, I/O devices, storage devices, etc., this provides the interface between computers and the real world systems.

Data

DBMS exists to collect, store, process and access data, the most important component. The database contains both the actual or operational data and the metadata.

Procedures

These are the instructions and rules that assist on how to use the DBMS, and in designing and running the database, using documented procedures, to guide the users that operate and manage it.

Database Access Language

This is used to access the data to and from the database, to enter new data, update existing data, or retrieve required data from databases. The user writes a set of appropriate commands in a database access language, submits these to the DBMS, which then processes the data and generates and displays a set of results into a user readable form.

Query Processor

This transforms the user queries into a series of low level instructions. This reads the online user's query and translates it into an efficient series of operations in a form capable of being sent to the run time data manager for execution.

Run Time Database Manager

Sometimes referred to as the database control system, this is the central software component of the DBMS that interfaces with user-submitted application programs and queries, and handles database access at run time. Its function is to convert operations in user's queries. It provides control to maintain the consistency, integrity and security of the data.

Data Manager

Also called the cache manger, this is responsible for handling of data in the database, providing a recovery to the system that allows it to recover the data after a failure.

Database Engine

The core service for storing, processing, and securing data, this provides controlled access and rapid transaction processing to address the requirements of the most demanding data consuming applications. It is often used to create relational databases for online transaction processing or online analytical processing data.

Data Dictionary

This is a reserved space within a database used to store information about the database itself. A data dictionary is a set of read-only table and views, containing the different information about the data used in the enterprise to ensure that database representation of the data follow one standard as defined in the dictionary.

Report Writer

Also referred to as the report generator, it is a program that extracts information from one or more files and presents the information in a specified format. Most report writers allow the user to select records that meet certain conditions and to display selected fields in rows and columns, or also format the data into different charts.

(OR)

b. Discuss different forms of SELECT with examples.

25. a. Explain 1 NF and 2 NF with a suitable examples.

(OR)

b. What is join operation? Explain different join operation with suitable table.

SQL Joins are used to relate information in different tables. A Join condition is a part of the sql query that retrieves rows from two or more tables. A SQL Join condition is used in the SQL WHERE Clause of select, update, delete statements.

The Syntax for joining two tables is:

```
SELECT col1, col2, col3...FROM table_name1, table_name2
WHERE table_name1.col2 = table_name2.col1;
```

If a sql join condition is omitted or if it is invalid the join operation will result in a Cartesian product. The Cartesian product returns a number of rows equal to the product of all rows in all the tables being joined. For example, if the first table has 20 rows and the second table has 10 rows, the result will be 20 * 10, or 200 rows. This query takes a long time to execute.

Lets use the below two tables to explain the sql join conditions.

database table "product";

product_id	product_name	supplier_name	unit_price
100	Camera	Nikon	300
101	Television	Onida	100
102	Refrigerator	Videocon	150
103	Ipod	Apple	75
104	Mobile	Nokia	50

database table "order_items";

order_id	product_id	total_units	customer
5100	104	30	Infosys
5101	102	5	Satyam
5102	103	25	Wipro
5103	101	10	TCS

SQL Joins can be classified into Equi join and Non Equi join.

1) SQL Equi joins

It is a simple sql join condition which uses the equal sign as the comparison operator. Two types of equi joins are SQL Outer join and SQL Inner join. For example: We can get the information about a customer who purchased a product and the quantity of product.

2) SQL Non equi joins

It is a sql join condition which makes use of some comparison operator other than the equal sign like >, <, >=, <=

1) SQL Equi Joins:

An equi-join is further classified into two categories:

a) SQL Inner Join

b) SQL Outer Join

a) SQL INNER Join or EQUI Join:

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Inner Join Syntax is,

```
SELECT column-name-list FROM
table-name1 INNER JOIN table-name2
WHERE table-name1.column-name = table-name2.column-name;
```

All the rows returned by the sql query satisfy the sql join condition specified.

For example: If we want to display the product information for each order the query will be as given below. Since you are retrieving the data from two tables, you need to identify the common column between these two tables, which is the product_id.

The query for this type of sql joins would be like,

```
SELECT order_id, product_name, unit_price, supplier_name, total_units FROM
product, order_items WHERE order_items.product_id = product.product_id;
```

The columns must be referenced by the table name in the join condition, because product_id is a column in both the tables and needs a way to be identified. This avoids ambiguity in using the columns in the SQL SELECT statement.

The number of join conditions is (n-1), if there are more than two tables joined in a query where 'n' is the number of tables involved. The rule must be true to avoid Cartesian product.

We can also use aliases to reference the column name, then the above query would be like,

```
SELECT o.order_id, p.product_name, p.unit_price, p.supplier_name,
o.total_units FROM product p, order_items o WHERE o.product_id =
p.product_id;
```

Example of INNER JOIN

Consider a **class** table,

ID NAME
1 abhi
2 adam
3 alex
4 anu

and the **class_info** table,

ID Address
1 DELHI
2 MUMBAI
3 CHENNAI

Inner JOIN query will be,

```
SELECT * from class INNER JOIN class_info where class.id = class_info.id;
```

The resultset table will look like,

ID NAME ID Address
1 abhi 1 DELHI
2 adam 2 MUMBAI
3 alex 3 CHENNAI

b)

SQL Self Join:

A Self Join is a type of sql join which is used to join a table to itself, particularly when the table has a FOREIGN KEY that references its own PRIMARY KEY. It is necessary to ensure that the join statement defines an alias for both copies of the table to avoid column ambiguity.

The below query is an example of a self join,

```
SELECT a.sales_person_id, a.name, a.manager_id, b.sales_person_id, b.name
FROM  sales_person a, sales_person b WHERE a.manager_id = b.sales_person_id;
```

2) SQL Non Equi Join:

A Non Equi Join is a SQL Join whose condition is established using all comparison operators except the equal (=) operator. Like >=, <=, <, >

For example: If you want to find the names of students who are not studying either Economics, the sql query would be like,

```
SELECT first_name, last_name, subject FROM student_details WHERE subject != 'Economics'
```

The output would be something like,

first_name	last_name	subject
Anjali	Bhagwat	Maths
Shekar	Gowda	Maths
Rahul	Sharma	Science
Stephen	Fleming	Science

We understand the benefits of taking a Cartesian product of two relations, which gives us all the possible tuples that are paired together. But it might not be feasible for us in certain cases to take a Cartesian product where we encounter huge relations with thousands of tuples having a considerable large number of attributes.

Join is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

We will briefly describe various join types in the following sections.

Theta (θ) Join

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol **θ**.

Notation

$R1 \bowtie_\theta R2$

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don't have anything in common, that is $R1 \cap R2 = \Phi$.

Theta join can use all kinds of comparison operators.

Student

SID	Name	Std
101	Alex	10
102	Maria	11

Subjects

Class	Subject

10 Math
10 English
11 Music
11 Sports

Student_Detail −

$$STUDENT \bowtie_{Student.Std = Subject.Class} SUBJECT$$

Student_detail

SID	Name	Std	Class	Subject
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

Equijoin

When Theta join uses only **equality** comparison operator, it is said to be equijoin. The above example corresponds to equijoin.

Natural Join (⋈)

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

The syntax for Natural Join is,

```
SELECT * FROM
table-name1 NATURAL JOIN table-name2;
```

Example of Natural JOIN

Here is the **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Natural join query will be,

```
SELECT * from class NATURAL JOIN class_info;
```

The resultset table will look like,

ID	NAME	Address
1	abhi	DELHI
2	adam	MUMBAI
3	alex	CHENNAI

In the above example, both the tables being joined have **ID** column(same name and same datatype), hence the records for which value of **ID** matches in both the tables will be the result of Natural Join of these two tables.

OUTER JOIN

This sql join condition returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables. The sql outer join operator in Oracle is (+) and is used on one side of the join condition only.

The syntax differs for different RDBMS implementation. Few of them represent the join conditions as "sql left outer join", "sql right outer join".

If you want to display all the product data along with order items data, with null values displayed for order items if a product has no order item, the sql query for outer join would be as shown below:

```
SELECT p.product_id, p.product_name, o.order_id, o.total_units FROM
order_items o, product p WHERE o.product_id (+) = p.product_id;
```

The output would be like,

product_id	product_name	order_id	total_units
100	Camera		
101	Television	5103	10
102	Refrigerator	5101	5
103	Ipod	5102	25

104 Mobile 5100 30

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

1. Left Outer Join
2. Right Outer Join
3. Full Outer Join

LEFT Outer Join

The left outer join returns a resultset table with the **matched data** from the two tables and then the remaining rows of the **left** table and null from the **right** table's columns.

Syntax for Left Outer Join is,

```
SELECT column-name-list FROM
table-name1 LEFT OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;
```

To specify a condition, we use the `ON` keyword with Outer Join.

Left outer Join Syntax for **Oracle** is,

```
SELECT column-name-list FROM
table-name1, table-name2 on table-name1.column-name = table-name2.column-
name(+);
```

Example of Left Outer Join

Here is the **class** table,

ID NAME
1 abhi
2 adam
3 alex
4 anu
5 ashish

and the **class_info** table,

ID Address
1 DELHI
2 MUMBAI

3 CHENNAI
7 NOIDA
8 PANIPAT

Left Outer Join query will be,

```
SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id = class_info.id);
```

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null

RIGHT Outer Join

The right outer join returns a resultset table with the **matched data** from the two tables being joined, then the remaining rows of the **right** table and null for the remaining **left** table's columns.

Syntax for Right Outer Join is,

```
SELECT column-name-list FROM
table-name1 RIGHT OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;
```

Right outer Join Syntax for **Oracle** is,

```
SELECT column-name-list FROM
table-name1, table-name2
ON table-name1.column-name(+) = table-name2.column-name;
```

Example of Right Outer Join

Once again the **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Right Outer Join query will be,

```
SELECT * FROM class RIGHT OUTER JOIN class_info ON (class.id =
class_info.id);
```

The resultant table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT

Full Outer Join

The full outer join returns a resultset table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

Syntax of Full Outer Join is,

```
SELECT column-name-list FROM
table-name1 FULL OUTER JOIN table-name2
ON table-name1.column-name = table-name2.column-name;
```

Example of Full outer join is,

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

and the **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Full Outer Join query will be like,

```
SELECT * FROM class FULL OUTER JOIN class_info ON (class.id = class_info.id);
```

The resultset table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

Reg.No:_____

[18CTU303]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established under Section 3 of UGC Act, 1956)

Coimbatore – 21

B.Sc Degree Examination

Computer Technology

Third Internal Examination – Oct 2019

Third Semester

Relational Database Management System

Class: II B.Sc CT **Duration : 2 Hours**

Date & Session: /10 /2019 & AN **Maximum : 50 Marks**

PART-A **[20 * 1 = 20 Marks]**

Answer ALL the questions

1. The _____ function is used to calculate the average value of the numeric type
   a) MIN b) AVG c) MAX d) COUNT
2. The _____statement is used to exit the loop from the reminder if its body either conditionally or unconditionally and forces the next iteration of the loop to take place, skipping any codes in between.
   a) break b) continue c) go to d) exit
3. PL/SQL _____is used when we want to execute a set of statements for a predetermined number of times.
   a) for loop b) while loop c) switch case d) if else
4. _____ is used to delete all the rows from the table and free the space containing the table.
   a) terminate b) delete c) drop d) truncate
5. The _____command is used to update or modify the value of a column in the table.
   a) truncate b) update c) insert d) delete
6. _____ is used to give user access privileges to a database.
   a) revoke b) savepoint c) commit d) grant
7. The _____used to roll the transaction back to a certain point without rolling back the entire transaction.
   a) commit b) rollback c) savepoint d) revoke
8. The SQL _____ statement is used to modify the data that is already in the database.
   a) UPDATE b) INSERT c) COMMIT d) ROLLBACK
9. _____is used to delete both the structure and record stored in the table.
   a) TRUNCATE b) DELETE c) ALTER d) DROP
10. The following one is not a DML command.
    a) INSERT b) UPDATE c) CREATE d) DELETE
11. _____ is used to search for values that are within a set of values
    a) BETWEEN b) AND c) EXISTS d) ANY
12. The following is not a TCL command.
    a) savepoint b) rollback c) commit d) revoke

13. _____ logical operator is used to search for the presence of a row in a specified table.
    a) EXISTS b) BETWEEN c) ANY d) IN
14. _____ function is used to calculate the sum of all selected columns.
    a) Count b) Max c) Min d) Sum
15. Pl/SQL stands for "Procedural Language extension of SQL" that is used in Oracle.
    a) Pro Language extension of SQL b) Procedural Language extension of SQL
    c) Programming Language extension of SQL d) Produce Language extension of SQL
16. _____ changes the structure of the table like creating a table, deleting a table, altering a table, etc.
    a) DML b) DCL c) TCL d) DCL
17. A _____ is a database object that groups related package constructs like Procedures, functions, cursor definitions, variables and constants, exception definitions.
    a) Package b) Trigger c) Cursor d) Exception
18. When an SQL statement is processed, Oracle creates a memory area known as context area and a _____ is a pointer to this context area.
    a) Trigger b) Exception c) Cursor d) Function
19. _____ is invoked by Oracle engine automatically whenever a specified event occurs.
    a) Exception b) Cursor c) Trigger d) Function
20. An error occurs during the program execution is called _____ in PL/SQL.
    a) Execution b) Bug c) Exception d) Cursor

PART-B **[3 * 2 = 6 Marks]**

Answer all of the following

21. What is DCL?
22. What are functions and procedures?
23. What ate packages?

PART-C **[3 * 8 = 24 Marks]**

Answer ALL of the following

24. a) Describe about DDL statements.

(OR)

b) Explain DML statements.

25. a) Write about Triggers.

(OR)

b) Explain about Cursors.

26. a) Explain about special operators for data access and aggregate functions.

(OR)

b) Discuss about PL/SQL.

Reg.No:_____

[18CTU303]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established under Section 3 of UGC Act, 1956)

Coimbatore – 21

B.Sc Degree Examination

Computer Technology

Third Internal Examination – Oct 2019

Third Semester

Relational Database Management System

Class: II B.Sc CT **Duration : 2 Hours**

Date & Session: /10 /2019 & AN **Maximum : 50 Marks**

PART-A **[20 * 1 = 20 Marks]**

Answer ALL the questions

1. The _____ function is used to calculate the average value of the numeric type
   a) MIN **b) AVG** c) MAX d) COUNT
2. The _____statement is used to exit the loop from the reminder if its body either conditionally or unconditionally and forces the next iteration of the loop to take place, skipping any codes in between.
   a) break **b) continue** c) go to d) exit
3. PL/SQL _____is used when we want to execute a set of statements for a predetermined number of times.
   a) for loop b) while loop c) switch case d) if else
4. _____ is used to delete all the rows from the table and free the space containing the table.
   a) terminate b) delete c) drop **d) truncate**
5. The _____command is used to update or modify the value of a column in the table.
   a) truncate **b) update** c) insert d) delete
6. _____ is used to give user access privileges to a database.
   a) revoke b) savepoint c) commit **d) grant**
7. The _____used to roll the transaction back to a certain point without rolling back the entire transaction.
   a) commit b) rollback **c) savepoint** d) revoke
8. The SQL _____ statement is used to modify the data that is already in the database.
   a) UPDATE b) INSERT c) COMMIT d) ROLLBACK
9. _____is used to delete both the structure and record stored in the table.
   a) TRUNCATE b) DELETE c) ALTER **d) DROP**
10. The following one is not a DML command.
    a) INSERT b) UPDATE c) CREATE d) DELETE
11. _____ is used to search for values that are within a set of values
    a) BETWEEN b) AND c) EXISTS d) ANY
12. The following is not a TCL command.
    a) savepoint b) rollback c) commit **d) revoke**

13. _____ logical operator is used to search for the presence of a row in a specified table.
a) EXISTS b) BETWEEN c) ANY d) IN
14. _____ function is used to calculate the sum of all selected columns.
a) Count b) Max c) Min **d) Sum**
15. Pl/SQL stands for _____ that is used in Oracle.
a) Pro Language extension of SQL **b) Procedural Language extension of SQL**
c) Programming Language extension of SQL d) Produce Language extension of SQL
16. _____ changes the structure of the table like creating a table, deleting a table, altering a table, etc.
a) DML b) DCL c) TCL **d) DDL**
17. A _____ is a database object that groups related package constructs like Procedures, functions, cursor definitions, variables and constants, exception definitions.
a) Package b) Trigger c) Cursor d) Exception
18. When an SQL statement is processed, Oracle creates a memory area known as context area and a _____ is a pointer to this context area.
a) Trigger b) Exception c) Cursor d) Function
19. _____ is invoked by Oracle engine automatically whenever a specified event occurs.
a) Exception b) Cursor **c) Trigger** d) Function
20. An error occurs during the program execution is called _____ in PL/SQL.
a) Execution b) Bug c) Exception d) Cursor

PART-B [3 * 2 = 6 Marks]

Answer all of the following

21. What is DCL?
A data control language (DCL) is a syntax similar to a computer programming language used to control access to data stored in a database (Authorization). In particular, it is a component of Structured Query Language (SQL). Examples of DCL commands include: GRANT to allow specified users to perform specified tasks.
22. What are functions and procedures?
"A procedures or function is a group or set of SQL and PL/SQL statements that perform a specific task." A function and procedure is a named PL/SQL Block which is similar . 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.
23. What ate packages?
"A package is a container for other database objects." A package can hold other database objects such as variables , consatants,cursors,exceptions,procedures,functions and sub-programs. ... A package's body fully defines cursors, functions, and procedures and thus implements the specification.

PART-C [3 * 8 = 24 Marks]

Answer ALL of the following

24. a) Describe about DDL statements.

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- CREATE - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- ALTER - alters the structure of the existing database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

(OR)

b) Explain DML statements.

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- SELECT - retrieve data from a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - Delete all records from a database table
- MERGE - UPSERT operation (insert or update)
- CALL - call a PL/SQL or Java subprogram
- EXPLAIN PLAN - interpretation of the data access path
- LOCK TABLE - concurrency Control

25. a) Write about Triggers.

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, 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 can be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes −

- Generating some derived column values automatically
- Enforcing referential integrity

- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is −

```
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;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name − Creates or replaces an existing trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will 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 will 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, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will 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.

Example

To start with, we will be using the CUSTOMERS table we had created and used in the previous chapters −

```
Select * from customers;
```

```
+----+----------+-----+-----------+----------+
| 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 |
+----+----------+-----+-----------+----------+
```

The following program creates 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;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

```
Trigger created.
```

(OR)

b) Explain about Cursors.

Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement; for example, the number of rows processed, etc.

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

- Implicit cursors
- Explicit cursors

Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has attributes such as **%FOUND, %ISOPEN, %NOTFOUND**, and **%ROWCOUNT**. The SQL cursor has additional attributes, **%BULK_ROWCOUNT** and **%BULK_EXCEPTIONS**, designed for use with the **FORALL** statement. The following table provides the description of the most used attributes −

S.No	Attribute & Description
1	**%FOUND** Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
2	**%NOTFOUND** The logical opposite of %FOUND. It returns TRUE if an INSERT,

	UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
3	**%ISOPEN** Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	**%ROWCOUNT** Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Any SQL cursor attribute will be accessed as **sql%attribute_name** as shown below in the example.

Example

We will be using the CUSTOMERS table we had created and used in the previous chapters.

```
Select * from customers;
```

```
+----+----------+-----+-----------+----------+
| 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 |
+----+----------+-----+-----------+----------+
```

The following program will update the table and increase the salary of each customer by 500 and use the **SQL%ROWCOUNT** attribute to determine the number of rows affected −

```
DECLARE
   total_rows number(2);
BEGIN
   UPDATE customers
   SET salary = salary + 500;
   IF sql%notfound THEN
      dbms_output.put_line('no customers selected');
   ELSIF sql%found THEN
```

```
      total_rows := sql%rowcount;
      dbms_output.put_line( total_rows || ' customers
selected ');
   END IF;
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

```
6 customers selected

PL/SQL procedure successfully completed.
```

If you check the records in customers table, you will find that the rows have been updated −

```
Select * from customers;

+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2500.00 |
|  2 | Khilan   |  25 | Delhi     |  2000.00 |
|  3 | kaushik  |  23 | Kota      |  2500.00 |
|  4 | Chaitali |  25 | Mumbai    |  7000.00 |
|  5 | Hardik   |  27 | Bhopal    |  9000.00 |
|  6 | Komal    |  22 | MP        |  5000.00 |
+----+----------+-----+-----------+----------+
```

26. a) Explain about special operators for data access and aggregate functions.

SQL Compound Operators

Operator Description

+= Add equals

-= Subtract equals

*= Multiply equals

/= Divide equals

%=Modulo equals

&= Bitwise AND equals

^-= Bitwise exclusive equals

|*= Bitwise OR equals

SQL Logical Operators

Operator Description Example

ALL TRUE if all of the subquery values meet the condition

AND TRUE if all the conditions separated by AND is TRUE

ANY TRUE if any of the subquery values meet the condition

BETWEEN TRUE if the operand is within the range of comparisons

EXISTS TRUE if the subquery returns one or more records

IN TRUE if the operand is equal to one of a list of expressions

LIKE TRUE if the operand matches a pattern

NOT Displays a record if the condition(s) is NOT TRUE

OR TRUE if any of the conditions separated by OR is TRUE

SOME TRUE if any of the subquery values meet the condition

An aggregate function allows you to perform a calculation on a set of values to return a single scalar value. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

The following are the most commonly used SQL aggregate functions:

- AVG – calculates the average of a set of values.
- COUNT – counts rows in a specified table or view.
- MIN – gets the minimum value in a set of values.
- MAX – gets the maximum value in a set of values.
- SUM – calculates the sum of values.

Notice that all aggregate functions above ignore `NULL` values except for the `COUNT` function.

SQL aggregate functions syntax

To call an aggregate function, you use the following syntax:

```
1    aggregate_function (DISTINCT | ALL expression)
```

Let's examine the syntax above in greater detail:

- First, specify an aggregate function that you want to use e.g., `MIN`, `MAX`, `AVG`, `SUM` or `COUNT`.
- Second, put `DISTINCT` or `ALL` modifier followed by an expression inside parentheses. If you explicitly use the `DISTINCT` modifier, the aggregate function ignores duplicate values and only consider the unique values. If you use the `ALL` modifier, the aggregate function uses all values for calculation or evaluation. The `ALL` modifier is used by default if you do not specify any modifier explicitly.

SQL aggregate function examples

Let's take a look some examples of using SQL aggregate functions.

COUNT function example

To get the number of products in the `products` table, you use the `COUNT` function as follows:

```
1    SELECT
2       COUNT(*)
3    FROM
4       products;
```

	COUNT(*)
▶	77

More information on the COUNT function.

AVG function example

To calculate the average units in stock of the products, you use the `AVG` function as follows:

```
1    SELECT
2       AVG(unitsinstock)
3    FROM
4       products;
```

	AVG(unitsinstock)
▶	40.5065

To calculate units in stock by product category, you use the `AVG` function with the `GROUP BY` clause as follows:

```
1    SELECT
2       categoryid, AVG(unitsinstock)
3    FROM
4       products
5    GROUP BY categoryid;
```

	categoryid	AVG(unitsinstock)
▶	1	46.5833
	2	42.2500
	3	29.6923
	4	39.3000
	5	44.0000
	6	27.5000
	7	20.0000
	8	58.4167

More information on AVG function.

SUM function example

To calculate the sum of units in stock by product category, you use the `SUM` function with the `GROUP BY` clause as the following query:

```
1    SELECT
2       categoryid, SUM(unitsinstock)
3    FROM
4       products
5    GROUP BY categoryid;
```

	categoryid	sum(unitsinstock)
▶	1	559
	2	507
	3	386
	4	393
	5	308
	6	165
	7	100
	8	701

Check it out the SUM function tutorial for more information on how to use the `SUM` function.

MIN function example

To get the minimum units in stock of products in the products table, you use the `MIN` function as follows:

```
1    SELECT
2        MIN(unitsinstock)
3    FROM
4        products;
```

	MIN(unitsinstock)
▶	0

More information on the MIN function.

MAX function example

To get the maximum units in stock of products in the products table, you use the `MAX` function as shown in the following query:

```
1    SELECT
2        MAX(unitsinstock)
3    FROM
4        products;
```

	Max(unitsinstock)
▶	125

Check it out the MAX function tutorial for more information.

In this tutorial, we have introduced you to the SQL aggregate functions including the most commonly used functions: `AVG`, `COUNT`, `MIN`, `MAX`, and `SUM`.

(OR)

b) Discuss about PL/SQL.

PL/SQL tutorial provides basic and advanced concepts of SQL. Our PL/SQL tutorial is designed for beginners and professionals.

PL/SQL is a block structured language that can have multiple blocks in it.

Our PL/SQL tutorial includes all topics of PL/SQL language such as conditional statements, loops, arrays, string, exceptions, collections, records, triggers, functions, procedures, cursors etc. There are also given PL/SQL interview questions and quizzes to help you better understand the PL/SQL language.

SQL stands for Structured Query Language i.e. used to perform operations on the records stored in database such as inserting records, updating records, deleting records, creating, modifying and dropping tables, views etc.

PL/SQL is a block structured language. The programs of PL/SQL are logical blocks that can contain any number of nested sub-blocks. Pl/SQL stands for "Procedural Language extension of SQL" that is used in Oracle. PL/SQL is integrated with Oracle database (since version 7). The functionalities of PL/SQL usually extended after each release of Oracle database. Although PL/SQL is closely integrated with SQL language, yet it adds some programming constraints that are not available in SQL.

PL/SQL Functionalities

PL/SQL includes procedural language elements like conditions and loops. It allows declaration of constants and variables, procedures and functions, types and variable of those types and triggers. It can support Array and handle exceptions (runtime errors). After the implementation of version 8 of Oracle database have included features associated with object orientation. You can create PL/SQL units like procedures, functions, packages, types and triggers, etc. which are stored in the database for reuse by applications.

With PL/SQL, you can use SQL statements to manipulate Oracle data and flow of control statements to process the data.

The PL/SQL is known for its combination of data manipulating power of SQL with data processing power of procedural languages. It inherits the robustness, security, and portability of the Oracle Database.