



KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established under section 3 of UGC Act 1956)

Coimbatore-641021

DEPARTMENT OF COMMERCE(UG)

Name: **G.Kowshalya, R.Lydia Priyadharsini, B. Shamile**

Department: **Commerce**

Subject Code: **17CCU402**

Semester: **IV**

Year: **2017-20 Batch**

Subject: **Database Management System- Lesson Plan**

UNIT - 1			
S. No	Lecture Hours	Contents	References
1.	1	Purpose of database - data independence, data model, view of data)	T1: Pg.No.: 2-4
2.	1	Overall system structure - query processor components, storage manager	T1: Pg.No.:16-18
3.	1	Entity Relationship Model - entities and entity sets, types of attributes, relationship	T1: Pg.No.: 23-28
4.	1	Mapping Constraints-one to one, one to many, many to one, many to many	T1: Pg.No.: 30-34
5.	1	Keys-Types of keys – primary key, super key, candidate key, foreign key	T1: Pg.No.:34-36
6.	1	E-R Diagrams	T1: Pg.No.:36-37
7.	1	Recapitulation and Discussion of Important Questions	-
Total no. of Hours planned for Unit 1			7
UNIT - 2			
1.	1	Relational Algebra – Introduction – characteristics of relational model	T1: Pg.No.: 63-64
2.	1	Structure – Basic structure, keys, super key, candidate & primary key	T1: Pg.No.: 64-71
3.	1	Formal query language – Procedural and Non Procedural query language)	T1:Pg.No.:71 w1,w2
4.	1	Relational algebra – select, project, Cartesian, product, rename, union)	T1: Pg.No.:71-86, W2
5.	1	Relational algebra – set intersection, assignment, division, join	W2
6.	1	Relational Calculus – Tuple Relational Calculus, Domain Relational Calculus	T1: Pg.No.:90-94
7.	1	Recapitulation and Discussion of Important Questions	-
Total no. of Hours planned for Unit 2			7
UNIT - 3			
1.	1	Structured query language	T1: Pg.No.:111- 113

2.	1	Basic structure – select clause, where clause, from clause . Rename operation, Tuple variables, String operations	T1: Pg.No.:113-120
3.	1	Set operations and Aggregate functions – Date, numeric, character functions	W1
4.	1	Nested sub queries – subqueries with the select statement – subqueries with insert, delete, update statements	T1: Pg.No.:125-129
5.	1	Modification of Databases	T1: Pg.No.:131-136
6.	1	Joined relations – join types and conditions, inner,outer,cross,union	T1: Pg.No.:136-146
7.	1	DDL[Data definition language]	T1: Pg.No.:140-145
8.	1	Embedded SQL	T1: Pg.No.:145-148
9	1	Recapitulation & Discussion on important questions	-
Total number of hours planned for Unit 3			9
UNIT - 4			
1.	1	Relational Database design	T1: Pg.No.:215
2.	1	Pitfalls – Redundance,waste space, null values	T1: Pg.No.:215-217
3.	1	Normalization using functional dependencies	W2
4.	1	First Normal form, Second Normal form	W1
5.	1	Third Normal form and Fourth Normal form	T1:Pg.No.: T1:228-231, w1
6.	1	Boyce Codd Normal form	W1
7.	1	Recapitulation and Discussion of Important Questions	-
Total no. of Hours planned for Unit 4			7
UNIT - 5			
1.	1	Oracle – Introduction – different editions of oracle, oracle database deployment	R2:Pg.No.: 3-15
2.	1	SQL- DML,DDL,DCL, TCL	R2:Pg.No.: 184-186
3.	1	Integrity Constraints –Types of Integrity Constraint(primary key, unique, foreign key, not null, check)	W3
4.	1	PL/SQL, PL/SQL block, PL/SQL Engine, syntax of PL/SQL, PL/SQL comments	R2:Pg.No.: 227-235
5.	1	Procedures in PL/SQL , Cursor Management	W3,W2 ,W4
6.	1	Triggers and Exception Handling	W4
7.	1	Recapitulation and Discussion of Important Questions	
8.	1	Revision of previous year question paper	
9.	1	Revision of previous year question paper	-
10	1	Revision of previous year question paper	-
Total no. of Hours planned for Unit 5			12

Suggested Readings:**Text Book**

1. Abraham Silber Sehatz, Henry F. Korth & S. Sudharasan. (2010) , *Database System Concepts*[7th Edition]. New Delhi, Tata McGraw Hill Mc Graw Hill Publication.

Reference Books

1. Singh. (2011) *Database systems: Concepts, Design & Applications*, [2nd Edition]. New Delhi, Pearson Education.
2. Gerald V.Post. (2011). *DBMS-Designing And Business Applications* [5th Edition] New Delhi, Tata McGraw Hill
3. Michael Abbey And Michael.J.Core.(2008). *Oracle- A Beginners guide* [4th Edition] New Delhi, Tata McGraw Hill.

Websites:

W1. www.tutorialpoint.com

W2: www.w3school.in

W3: www.javapoint.com/sql-tutorial

W3: www.tutorailpointexamples.com/sql

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

UNIT I

SYLLABUS

Purpose of Database - Overall System Structure - Entity Relationship Model - Mapping Constraints - Keys - E-R Diagrams.

INTRODUCTION TO DATABASE SYSTEMS:

Data

- The raw facts are called as data.
- The word “raw” indicates that they have not been processed.

Ex: For example 89 is the data.

Information

- The processed data is known as information.

Ex: Marks: 89; then it becomes information.

Knowledge

- Knowledge refers to the practical use of information.
- Knowledge necessarily involves an experience.

DATA/INFORMATION PROCESSING:

The process of converting the data (raw facts) into meaningful information is called as data/information processing.



DIFFERENCE BETWEEN DATA AND INFORMATION:

S.No	DATA	INFORMATION
1	Raw facts	Processed data
2	It is in unorganized form	It is in organized form
3	Data doesn't help in decision making process	Information helps in decision making process

DATABASE

- A database is a collection of inter-related data.
- A database is a collection of information that is organized so that it can be easily accessed, managed and updated.

For example,

Consider data of bank accounts. Here, all the data - id, name, address, and contact no - interrelated. So, they represent some particular customer.

Applications of Database Systems

The following are the various kinds of applications/organizations uses databases for their business processing activities in their day-to-day life.

They are:

1. Banking: For customer information, accounts, and loans, and banking transactions.
2. Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner—terminals situated around the world accessed the central database system through phone lines and other data networks.
3. Universities: For student information, course registrations, and grades.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

4. Credit Card Transactions: For purchases on credit cards and generation of monthly statements.
5. Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
6. Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
7. Sales: For customer, product, and purchase information.
8. Manufacturing: For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.
9. Human resources: For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.
10. Railway Reservation Systems: For reservations and schedule information.
11. Web: For access the Bank accounts and to get the balance amount.
12. E –Commerce: For Buying a book or music CD and browse for things like watches, mobiles from the Internet.

DATABASE MANAGEMENT SYSTEM

- A database management system (DBMS) is a collection of inter-related data and a set of programs to manipulate those data.
- Database management system (DBMS) stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.
- Data manipulation consist various operation like store data, modify data, remove data, retrieve that data.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

- So, DBMS = Database + Set of programs
- It also provides security or safety against system crashes.

Goals of DBMS

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

- Manage large bodies of information.
- Provide convenient and efficient ways to store and access information.
- Secure information against system failure or corrupt.
- Permit data to be shared among multiple users.

PURPOSE OF DATABASE SYSTEM

The typical file processing system is supported by a conventional operating system. Permanent records are stored in various files and a number of different application programs are written to extract records from and add records to the appropriate files.

DISADVANTAGES OF FILE PROCESSING SYSTEM:

The following are the disadvantages of File Processing System:

Data Redundancy:

- The same information may be duplicated in several places with different format. This redundancy leads to higher storage and access cost.
- Data Inconsistency:
- Because of data redundancy, it is possible that data may not be in consistent state.

Difficulty in Accessing Data:

- Accessing data is not convenient and efficient in file processing system.

Data Isolation Or Limited Data Sharing::

- Data are scattered in various files.
- Also different files may have different formats and these files may be stored in different folders may be of different departments.
- So, due to this data isolation, it is difficult to share data among different applications.

Integrity Problems:

- Data integrity means that the data contained in the database in both correct and consistent.
- For this purpose the data stored in database must satisfy correct and constraints.

Atomicity Problems:

- Any operation on database must be atomic.
- This means, it must happen in its entirely or not at all.

Concurrent Access Anomalies:

- Multiple users are allowed to access data simultaneously. This is for the sake of better performance and faster response.

Security Problems:

- Database should be accessible to users in limited way.
- Each user should be allowed to access data concerning his requirements only.

For Example

Consider part of a savings-bank enterprise that keeps information about all customers and savings accounts. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

information, the system has a number of application programs that manipulate the files, including:

- A program to debit or credit an account
- A program to add a new account
- A program to find the balance of an account
- A program to generate monthly statements

Programmers wrote these application programs to meet the needs of the bank.

New application programs are added to the system as the need arises.

For example, suppose that the savings bank decides to offer checking accounts.

As a result, the bank creates new permanent files that contain information about all the checking accounts maintained in the bank, and it may have to write new application programs to deal with situations that do not arise in savings accounts, such as overdrafts.

Thus, as time goes by, the system acquires more files and more application programs.

The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files.

Before database management systems (DBMS) came along, organizations usually stored information in file-processing system.

Organizational information in a file-processing system has a number of major disadvantages:

Data Redundancy and Inconsistency:

The address and telephone number of a particular customer may appear in a file that consists of savings-account records and in a file that consists of checking-account records. This redundancy leads to higher storage and access cost. In, it

may lead to data inconsistency; that is, the various copies of the same data may no longer agree. For example, a changed customer address may be reflected in savings-account records but not elsewhere in the system.

Difficulty in Accessing Data:

Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. The officer asks the data-processing department to generate such a list. Because there is no application program to generate that. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory.

Data Isolation:

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

Integrity Problems:

The balance of a bank account may never fall below a prescribed amount (say, \$25). Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

Atomicity Problems:

A computer system, like any other mechanical or electrical device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. Consider a

program to transfer \$50 from account A to account B. If a system failure occurs during the execution of the program, it is possible that the \$50 was removed from account A but was not credited to account B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be atomic—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

Concurrent-Access Anomalies:

For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data. Consider bank account A, containing \$500. If two customers withdraw funds (say \$50 and \$100 respectively) from account A at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value \$500, and write back \$450 and \$400, respectively. Depending on which one writes the value last, the account may contain \$450 or \$400, rather than the correct value of \$350. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

Security Problems:

Not every user of the database system should be able to access all the data. For example, in a banking system, payroll personnel need to see only that part of the

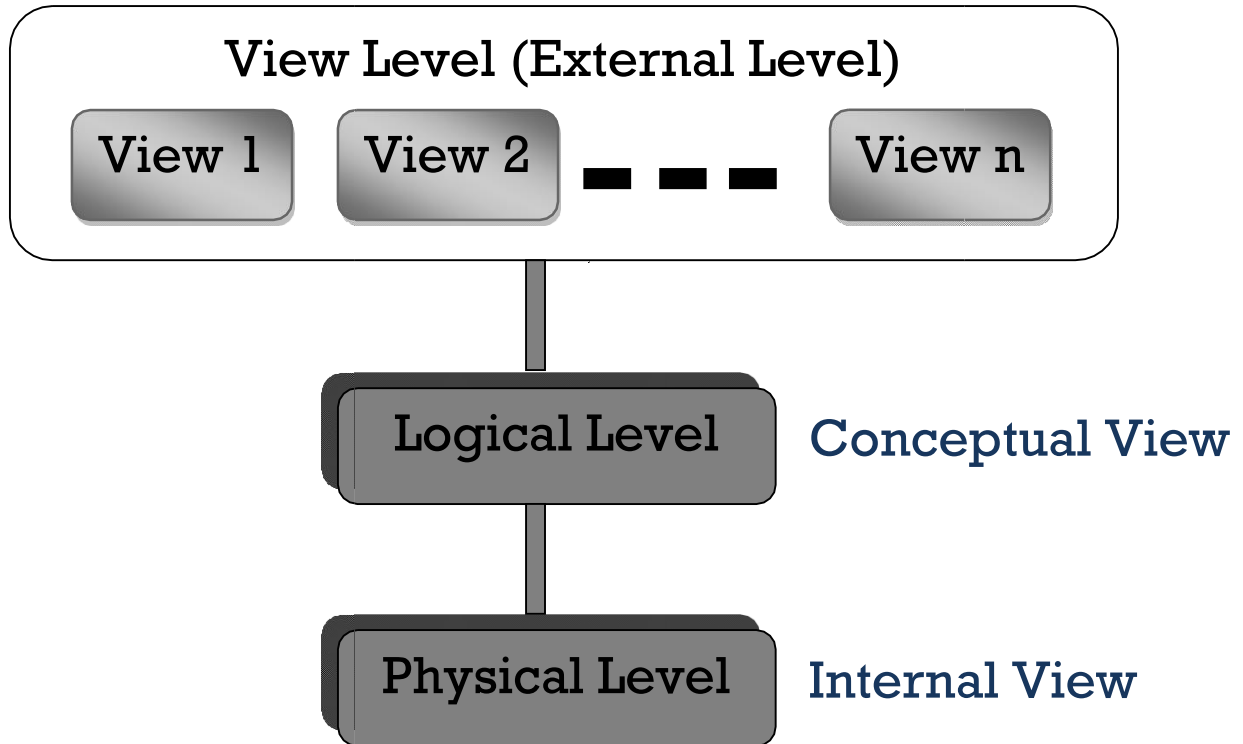
database that has information about the various bank employees. They do not need access to information about customer accounts. But, since application programs are added to the system in an ad hoc manner, enforcing such security constraints is difficult. These difficulties, among others, prompted the development of database systems.

VIEW OF DATA

- A DBMS is a collection of interrelated files and a set of programs that allow users to access and modify these files.
- A major purpose of a database system is to provide users with an abstract view of the data.
- i.e., the system hides certain details of how the data are stored and maintained.
- To understand the view of data, we must have a basic knowledge of
 - i) Data Abstraction and ii) Instance & Schema.

Data Abstraction

- The major purpose of a database system is to provide users with an **abstract view** of the system.
- The system hides certain details of how data is stored and created and maintained
- Complexity should be hidden from database users.



1. Physical Level or Internal View:

- This is the lowest level of data abstraction.
- It describes how the data (e.g., Customer) are actually stored on storage devices (for example words or bytes).
- It is also known as physical level.
- It provides internal view of physical storage of data.
- It deals with complex low level data structures, file structures and access methods in detail.
- It also deals with Data Compression and Encryption techniques, if used.

2. Logical Level or Conceptual View:

- This is the next higher level than internal level of data abstraction.
- It describes what data are stored in the database and what relationships exist among those data.

type *customer* = **record**

name : string; *street* : string; *city*

: integer;

end;

- It is also known as logical level.
- It hides low level complexities of physical storage.
- Database administrator and designers work at this level to determine what data to keep in database.
- Application developers also work on this level.

3. View Level or External Level:

- This is the highest level of data abstraction.
- It describes only part of the entire database that an end user concern.
- It is also known as a view level.
- End users need to access only part of the database rather than entire database.
- Different user needs different views of database. And so, there can be many view level abstractions of the same database.

For example, tellers in a bank see only part of the information on customer accounts; they cannot access information concerning salaries of employees.

Instances and Schemas

- Schema – the logical structure of the database (e.g., set of customers and accounts and the relationship between them)
- Instance: The collection of information stored in the database at a particular moment.

The difference between Schema and Instance are given in below table.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

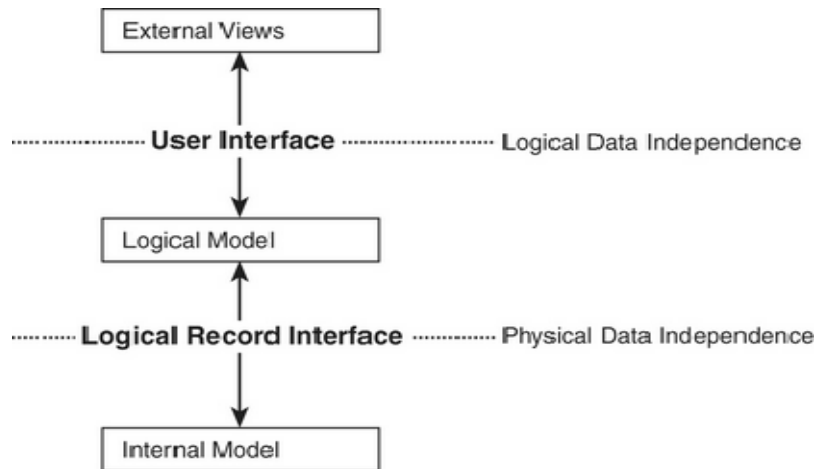
	Schema	Instance
Definition	The overall logical design of the database.	The collection of information stored in the database at a particular moment.
Includes	Table names, column names, data types, and size of columns, various constraint at logical level.	Actual data or information stored in tables in form of different records.
Change	Changes infrequently.	Changes frequently.
Cause of change	Insertion of tables or columns and change in datatype,size or constraints on any column.	insert, delete or update operation on data stored in database.
Analogy	variable declaration	value of the variable

Data Independence

- Data independence is ability to modify a schema definition in one level without affecting a schema definition in the next higher level.

There are two levels of data independence:

1. Physical Data Independence
2. Logical Data Independence



Physical Data Independence

- It is the ability to modify the physical schema without requiring any change in application programs.
- Modifications at the internal levels are occasionally necessary to improve performance.
- Possible modifications at internal levels are change in file structures, compression techniques, hashing algorithms, storage devices, etc.
- Physical data independence separates conceptual levels from the internal levels.
- This allows providing a logical description of the database without the need to specify physical structures.
- Comparatively, it is easy to achieve physical data independence.

Logical Data Independence

- It is ability to modify the conceptual schema without requiring any change in application programs.
- Modifications at the logical levels are necessary whenever the logical structure of the database is altered.
- Logical data independence separates external level from the conceptual

view.

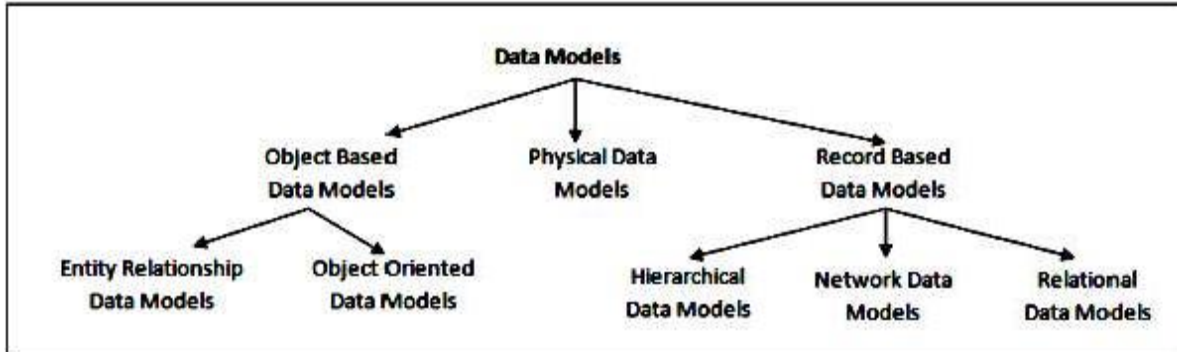
- Comparatively it is difficult to achieve logical data independence.
- Application programs are heavily dependent on logical structures of the data they access. So any change in logical structure also requires programs to change.

Difference between Logical Data Independence & Physical Data Independence

Logical Data Independence	Physical Data Independence
It is concerned with the structure of the data or changing the data definition.	It is concerned with storage of the data.
It is very difficult as the retrieving of data are heavily dependent on logical	It is easy to retrieve.
Application program need not be changed if new fields are added or deleted from the database.	Physical database is concerned with the change of the storage device.
It is concerned with the conceptual schema.	It is concerned with the internal schema.

DATA MODELS

- The structure of the database is called the data models.
- A Collection of conceptual tools for describing data, data relationship, data semantic and consistency constraint.



Object based Data Models

- It is designed using the entities in the real world, attributes of each entity and their relationship.
- It picks up each thing/object in the real world which is involved in the requirement.

There are two types of object based data Models

- ❖ Entity Relationship Model
- ❖ Object oriented data model.

ER data model is one of the important data model which forms the basis for the all the designs in the database world. It defines the mapping between the entities in the database. Object oriented data model, along with the mapping between the entities, describes the state of each entity and the tasks performed by them.

Entity-Relationship Model

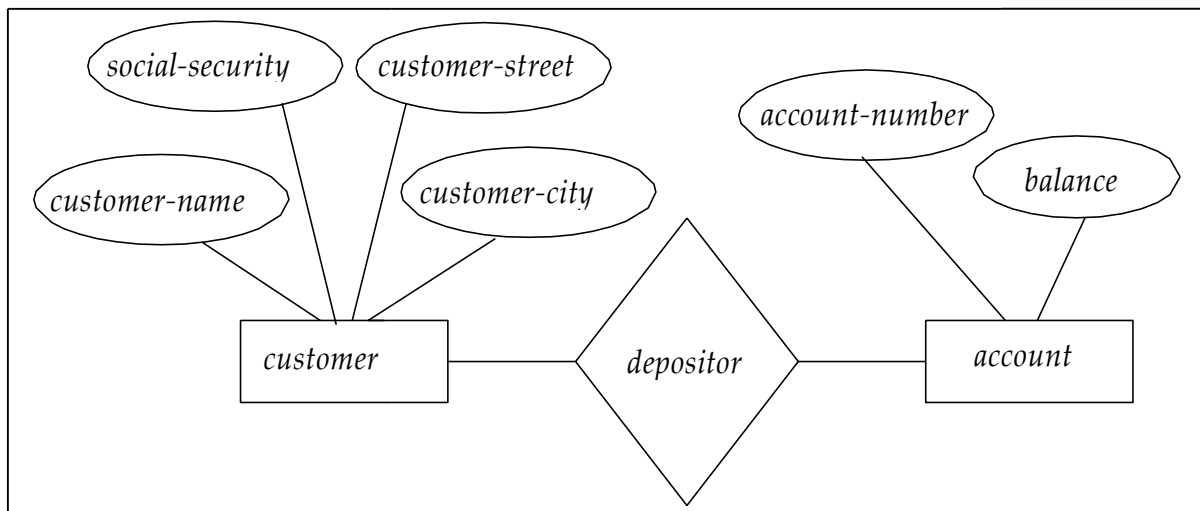
- The entity-relationship model is based on a perception of the world as consisting of a collection of basic objects (entities) and relationships among these objects.
- An entity is a distinguishable object that exists.
- Each entity has associated with it a set of attributes describing it.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

- E.g. number and balance for an account entity.
- A relationship is an association among several entities.
- E.g. depositor relationship associates a customer with each account he or she has.
- The set of all entities or relationships of the same type is called the entity set or relationship set.
- Another essential element of the E-R diagram is the mapping cardinalities, which express the number of entities to which another entity can be associated via a relationship set.
- The overall logical structure of a database can be expressed graphically by an E-R diagram:
 - ❖ Rectangles: represent entity sets.
 - ❖ Ellipses: represent attributes.
 - ❖ Diamonds: represent relationships among entity sets.
 - ❖ Lines: link attributes to entity sets and entity sets to relationships.



Object-Oriented Database Model

- In the object oriented data model the (OODM). Both data and their relationship are contained in a single structure known as an object.
- An object includes information about relationship between the facts within the object, as well as information about its relationship with other objects.
- It is said to be “SEMANTIC DATA MODEL”

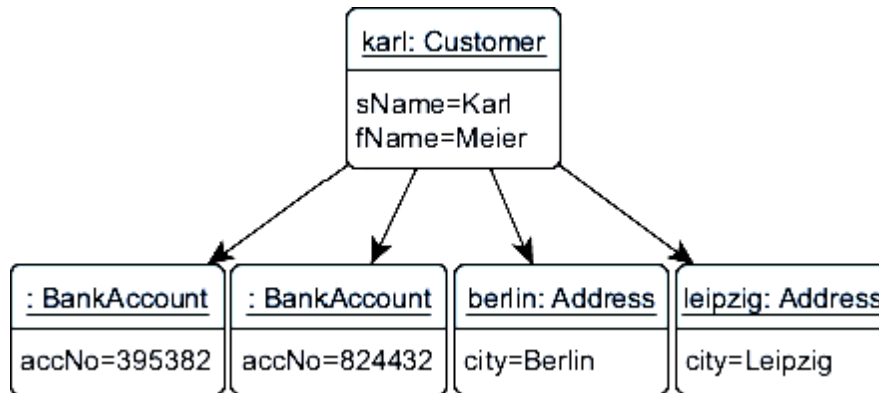
Object Oriented Data model components

- An object is the abstraction of the real- world entity. An object represents only one occurrence of entity.
- Attributes describe the property of an object.
- Objects that are similar in characteristics are grouped in class.
- **Class:** is a collection of similar objects with shared structure (attributes) and behavior (method)
- **Method:** represents a real word action such as finding a selected person's name, changing person's name or printing a person's address.
- Classes are organized in class hierarchy. The class hierarchy resembles an upside down tree in which each class has only one parent.
- Inheritance is the ability of an object within the class hierarchy to inherit the attributes and methods of the class.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020



Relational Model

- The relational model uses a collection of tables to represent both data and the relationships among those data.
- Each table has multiple columns, and each column has unique name.

Customer name	Social security	Customer street	Customer city	Account number
Johnson	192-83-7465	Alma	Palo Alto	A-101
Smith	019-28-3746	North	Rye	A-215
Johnson	192-83-7465	Alma	Palo Alto	A-201
Jones	321-12-3123	Main	Harrison	A-217
Smith	019-28-3746	North	Rye	A-201

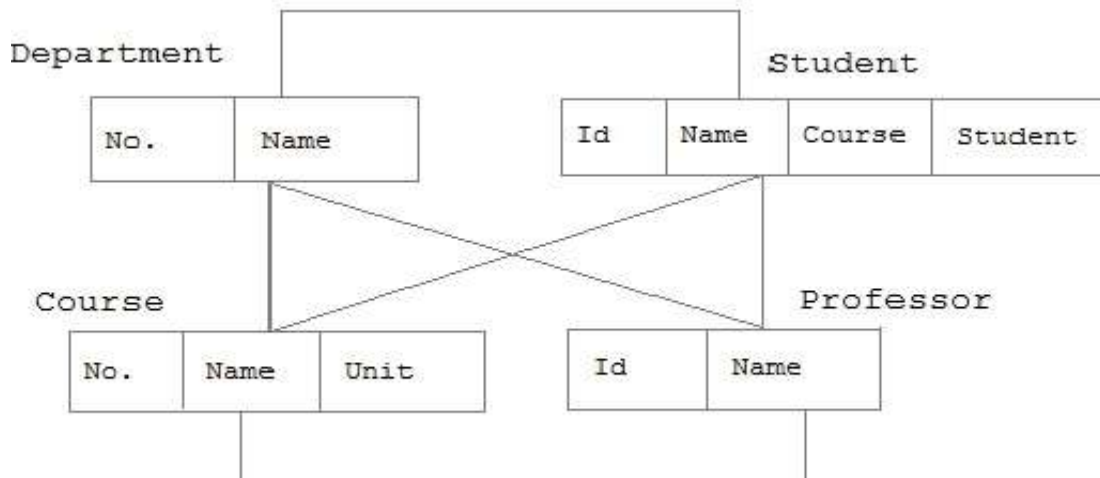
Network Model

- Data are represented by collections of records.
- Relationships among data are represented by links.
- Organization is that of an arbitrary graph.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

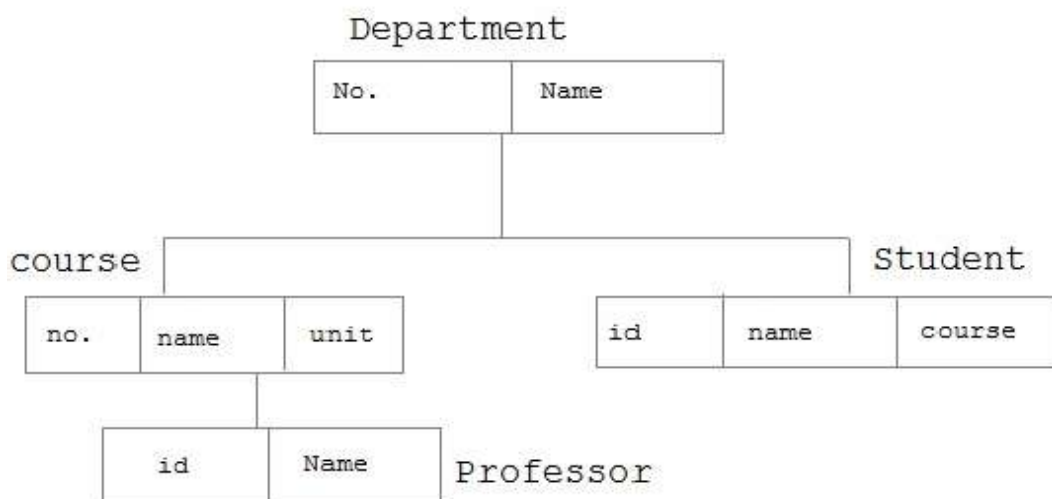
COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020



Hierarchical Model

- In hierarchical model the data and relationships among the data are represented by records and links.
- It is same as network model but differs in terms of organization of records as collections of trees rather than graphs.



DBMS Languages

- A Database system provides two different types of languages one to specify the database schema and other to express database queries and updates.

Data-Definition Languages

- Specification notation for defining the database schema
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
- *Data storage and definition* language – special type of DDL in which the storage structure and access methods used by the database system are specified.

Data Manipulation Language

- Data Manipulation is:
 - ❖ Retrieval of information from the database
 - ❖ Insertion of new information into the database
 - ❖ Deletion of information in the database
 - ❖ Modification of information in the database
- A DML is a language which enables users to access and manipulate data.
- The goal is to provide efficient human interaction with the system.

There are two types of DML:

- procedural: the user specifies what data is needed and how to get it
 - nonprocedural: the user only specifies what data is needed
 - ❖ Easier for user.
 - ❖ May not generate code as efficient as that produced by procedural languages.
 - A query language is a portion of a DML involving information retrieval only.
- The terms DML and query language are often used synonymously.

COMPONENTS OF DBMS

- Basic Components: A database system has four components.
- These four components are important for understanding and designing the database system.

These are:

1. Data
2. Hardware
3. Software
4. Users

1. Data

- Data is raw hand information collected by us.
- It may consist of bits or bytes.
- A Data item is often referred to as field or data element.
- A Data aggregate is the collection of data items within the record, which is given a name and referred as a whole.
- Data can be collected orally or written.
- A database can be integrated and shared.

2. Hardware

- Hardware is also a major and primary part of the database.
- Without hardware nothing can be done

3. Software

- Software is another major part of the database system.
- Software is a system. Software is further subdivided into two categories.
- First type is system software (like all the operating systems, all the languages and system packages etc.)
- Second is application software (payroll, electricity billing, hospital management and hostel administration etc.).

- By using software, data can be manipulated, organized and stored.

4. Users

- User can collect the data, operate and handle the hardware.
- Also operator feeds the data and arranges the data in order by executing the software.

TRANSACTION MANAGEMENT

- A *transaction* is a collection of operations that performs a single logical function in a database application.
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g. power failures and operating system crashes) and transaction failures.
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

STORAGE MANAGEMENT

- A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for the following tasks:
 - ❖ interaction with the file manager
 - ❖ efficient storing, retrieving, and updating of data

DATABASE ADMINISTRATOR

- The database administrator is a person having central control over data and programs accessing that data.
- Functions of the database administrator include:

- ❖ **Scheme definition:** the creation of the original database scheme. This involves writing a set of definitions in a DDL (data storage and definition language), compiled by the DDL compiler into a set of tables stored in the data dictionary.
- ❖ **Storage structure and access method definition:** writing a set of definitions translated by the data storage and definition language compiler.
- ❖ **Scheme and physical organization modification:** writing a set of definitions used by the DDL compiler to generate modifications to appropriate internal system tables (e.g. data dictionary). This is done rarely, but sometimes the database scheme or physical organization must be modified.
- ❖ **Granting of authorization for data access:** granting different types of authorization for data access to various users
- ❖ **Integrity constraint specification:** generating integrity constraints. These are consulted by the database manager module whenever updates occur.

DATABASE USERS

The database users fall into several categories:

- **Application programmers** are computer professionals interacting with the system through DML calls embedded in a program written in a host language (e.g. C, PL/1, Pascal).
 - ❖ These programs are called application programs.
 - ❖ The DML precompiler converts DML calls (prefaced by a special character like \$, #, etc.) to normal procedure calls in a host language.
 - ❖ The host language compiler then generates the object code.
 - ❖ Some special types of programming languages combine Pascal-like

control structures with control structures for the manipulation of a database.

- ❖ These are sometimes called fourth-generation languages.
- ❖ They often include features to help generate forms and display data.
- **Sophisticated users** interact with the system without writing programs.
 - ❖ They form requests by writing queries in a database query language.
 - ❖ These are submitted to a query processor that breaks a DML statement down into instructions for the database manager module.
- **Specialized users** are sophisticated users writing special database application programs. These may be CADD systems, knowledge-based and expert systems, complex data systems (audio/video), etc.
- **Naive users** are unsophisticated users who interact with the system by using permanent application programs (e.g. automated teller machine).

Overall System Structure

Components of DBMS are broadly classified as follows:

1. Query Processor Components:

- **DML Pre-compiler:** It translates DML statements in a query language into low level instructions that query evaluation engine understand. It also attempts to transform user's request into an equivalent but more efficient form.
- **Embedded DML Pre-compiler:** It converts DML statements embedded in an application program to normal procedure calls in the host language. The Pre-compiler must interact with the DML compiler to generate the appropriate code.
- **DDL Interpreter:** It interprets the DDL statements and records them in a set of tables containing metadata or data dictionary.
- **Query Evaluation Engine:** It executes low-level instructions generated by

the DML compiler.

2. Storage Manager Components:

They provide the interface between the low-level data stored in the database and application programs and queries submitted to the system.

- Authorization and Integrity Manager: It tests for the satisfaction of integrity constraints checks the authority of users to access data.
- Transaction Manager: It ensures that the database remains in a consistent state despite the system failures and that concurrent transaction execution proceeds without conflicting.
- File Manager: It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- Buffer Manager: It is responsible for fetching data from disk storage into main memory and deciding what data to cache in memory.

3. Data Structures:

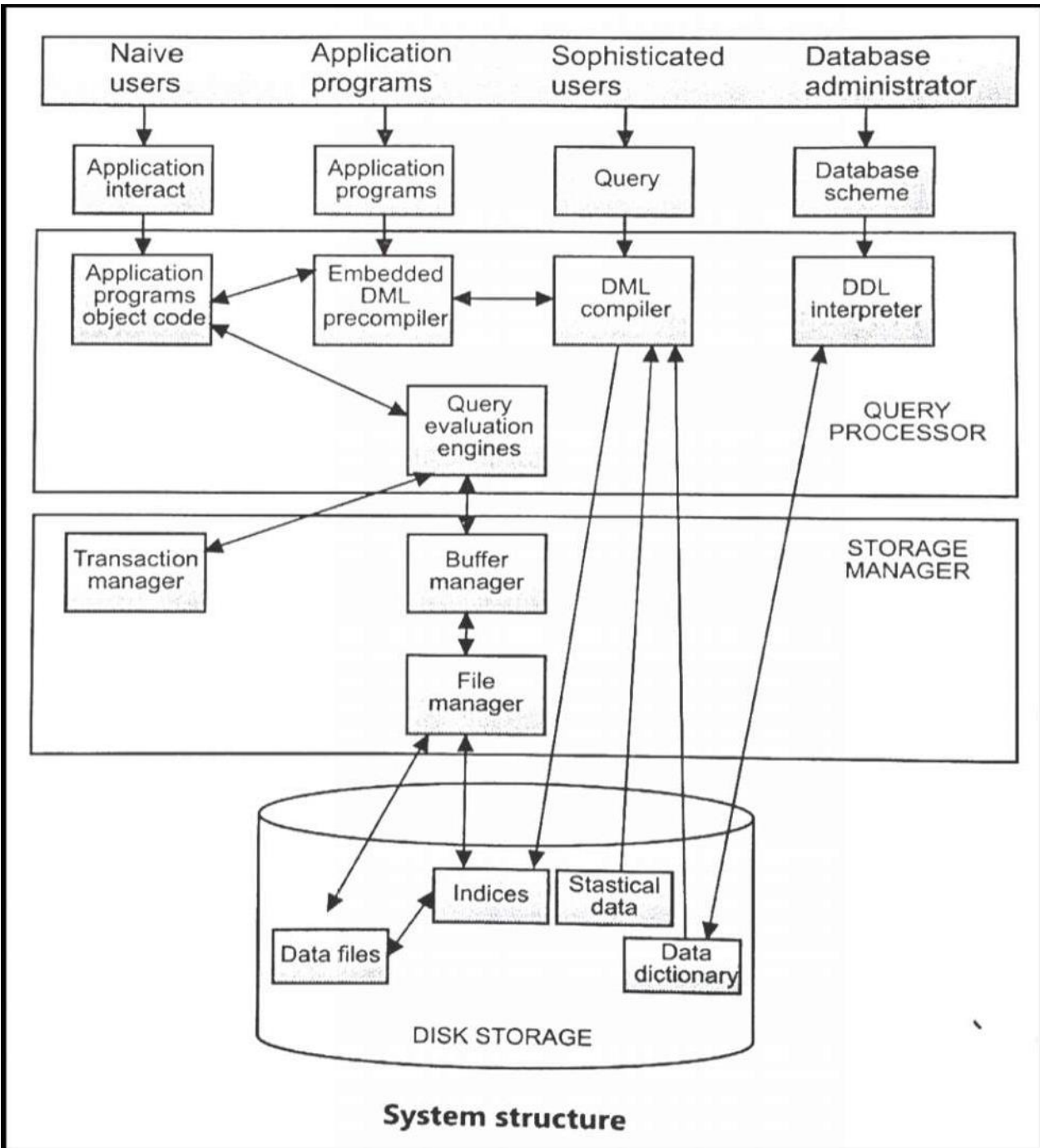
Following data structures are required as a part of the physical system implementation.

- Data Files: It stores the database.
- Data Dictionary: It stores meta data (data about data) about the structure of the database.
- Indices: Provide fast access to data items that hold particular values.
- Statistical Data: It stores statistical information about the data in the database. This information is used by query processor to select efficient ways to execute query.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

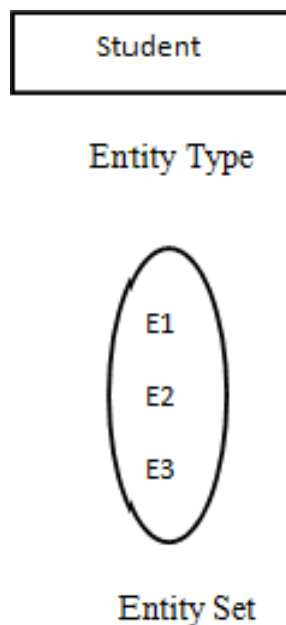


THE ENTITY-RELATIONSHIP MODEL

- The E-R (entity-relationship) data model views the real world as a set of basic objects (entities) and relationships among these objects.

Entities and Entity Sets

- An entity is an object that exists and is distinguishable from other objects.
For example: specific person, company, event, plant
- An entity set is a set of entities of the same type.
For example: set of all persons, companies, trees, holidays
- In ER diagram, Entity and Entity Set is represented as:



- An entity is represented by a set of attributes, that is, descriptive properties possessed by all members of an entity set.

For example:

(Entity) customer = **attributes are** (customer-name, social-security,

CLASS: II B.COM CA
COURSE CODE: 17CCU402

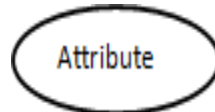
COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

customer-street, customer-city)

(Entity) account = **attributes are** (account-number, balance)

- In ER diagram, attribute is represented by an oval.



- Domain – the set of permitted values for each attribute (e.g. the telephone number must be seven positive integers).
- A domain is defined as the set of all unique values permitted for an attribute.
- Each table has certain set of columns and each column allows a same type of data, based on its data type. The column does not accept values of any other data type.
- For example, a domain of date is the set of all possible valid dates, a domain of integer is all possible whole numbers, a domain of day-of-week is Monday, Tuesday ... Sunday.

Types of Attributes

The different types of attributes are as follows

- Single valued attributes
- Multi valued attributes
- Compound /Composite attributes
- Simple / Atomic attributes
- Stored attributes
- Derived attributes
- Complex attributes
- Key attributes

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

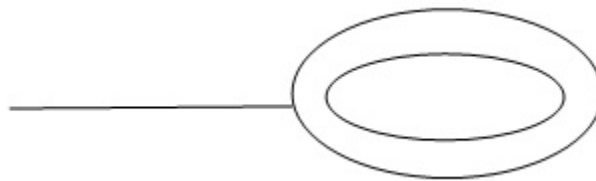
- Non key attributes
- Required attributes
- Optional/ null value attributes

Single Valued Attributes: It is an attribute with only one value.

- Example: Any manufactured product can have only one serial no. , but the single valued attribute cannot be simple valued attribute because it can be subdivided. Likewise in the above example the serial no. can be subdivided on the basis of region, part no. etc.

Multi Valued Attributes: These are the attributes which can have multiple values for a single or same entity.

- Example: Car's colors can be divided into many colors like for roof, trim.
- The notation for multi valued attribute is:



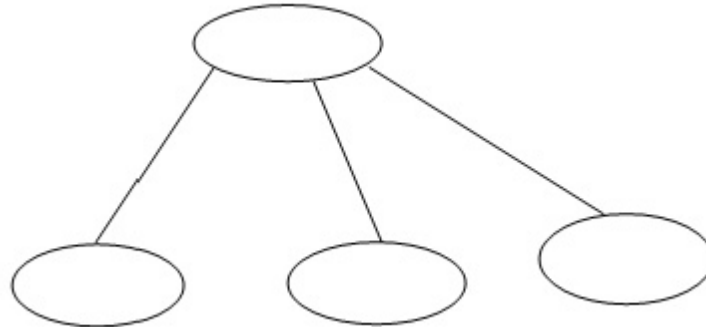
Compound / Composite attributes: This attribute can be further divided into more attributes.

- The notation for it is:

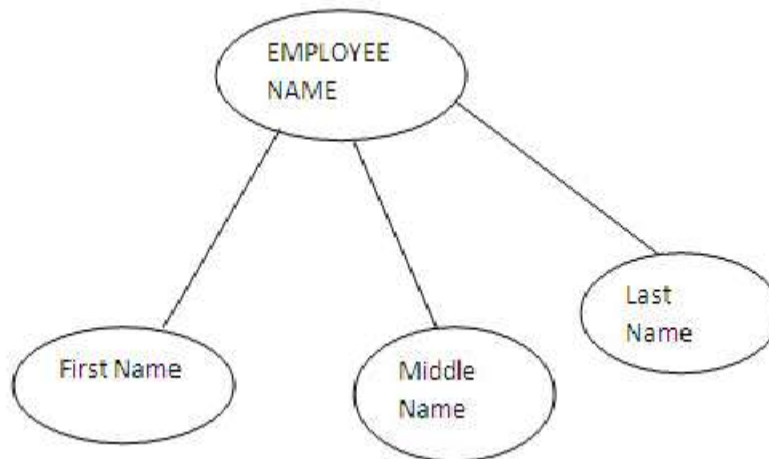
CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020



- Example: Entity Employee Name can be divided into sub divisions like FName, MName, LName.



Simple / Atomic Attributes: The attributes which cannot be further divided are called as simple / atomic attributes.

- Example: The entities like age, marital status cannot be subdivided and are simple attributes.

Stored Attributes: Attribute that cannot be derived from other attributes are called as stored attributes.

- Example: Birth date of an employee is a stored attribute.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

Derived Attributes: These attributes are derived from other attributes. It can be derived from multiple attributes and also from a separate table.

- Example: Today's date and age can be derived. Age can be derived by the difference between current date and date of birth.
- The notation for the derived attribute is:

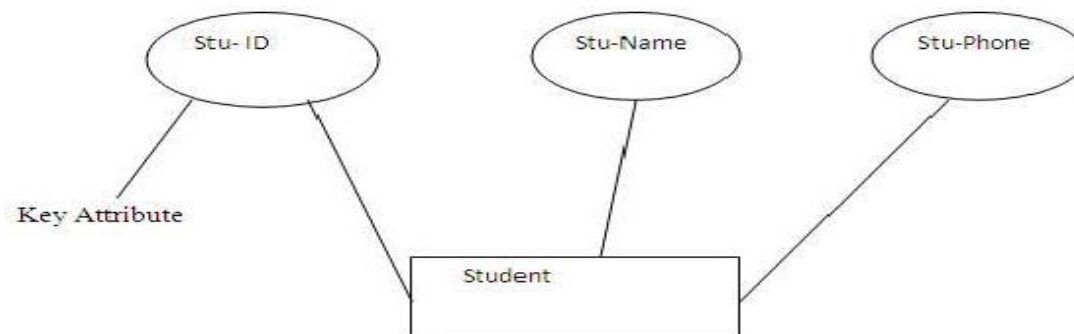


Complex Attributes: For an entity, if an attribute is made using the multi valued attributes and composite attributes then it is known as complex attributes.

- Example: A person can have more than one residence; each residence can have more than one phone.

Key Attributes: This attribute represents the main characteristic of an entity i.e. primary key. Key attribute has clearly different value for each element in an entity set.

- Example: The entity student ID is a key attribute because no other student will have the same ID.



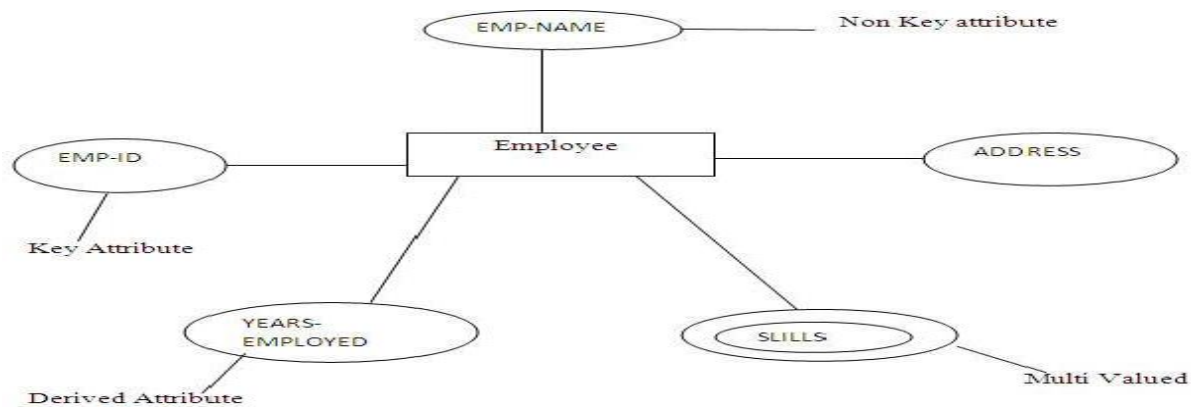
CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

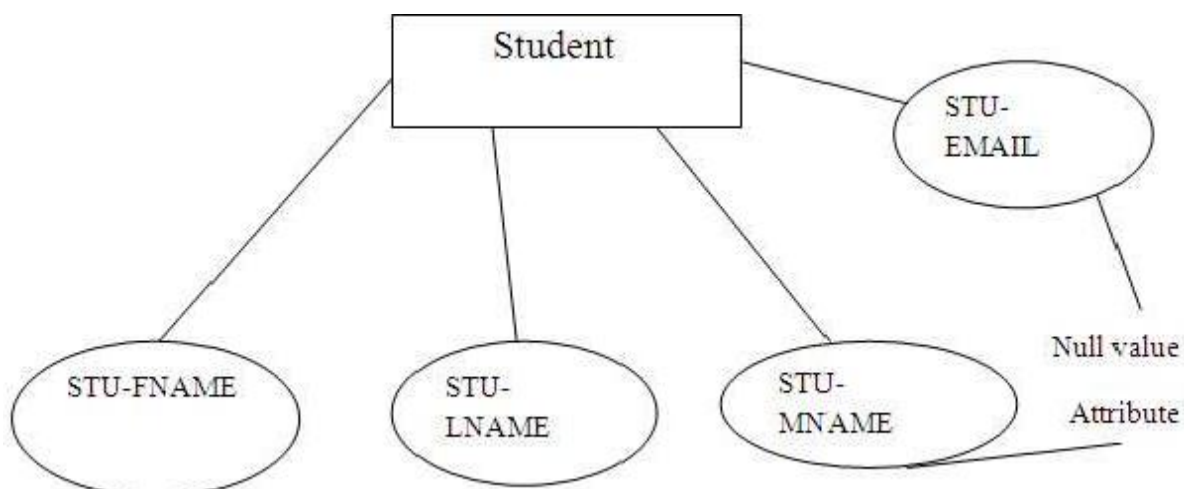
Non- Key Attributes: Excluding the candidate key attributes in an entity set are the non key attributes.

- Example: First name of a student or employee is a non-key attribute as it does not represent main characteristic of an entity.



Optional / Null value Attributes: It does not have a value and can be left blank, it's optional can be filled or cannot be.

- Example: Considering the entity student there the student's middle name and the email ID is optional.



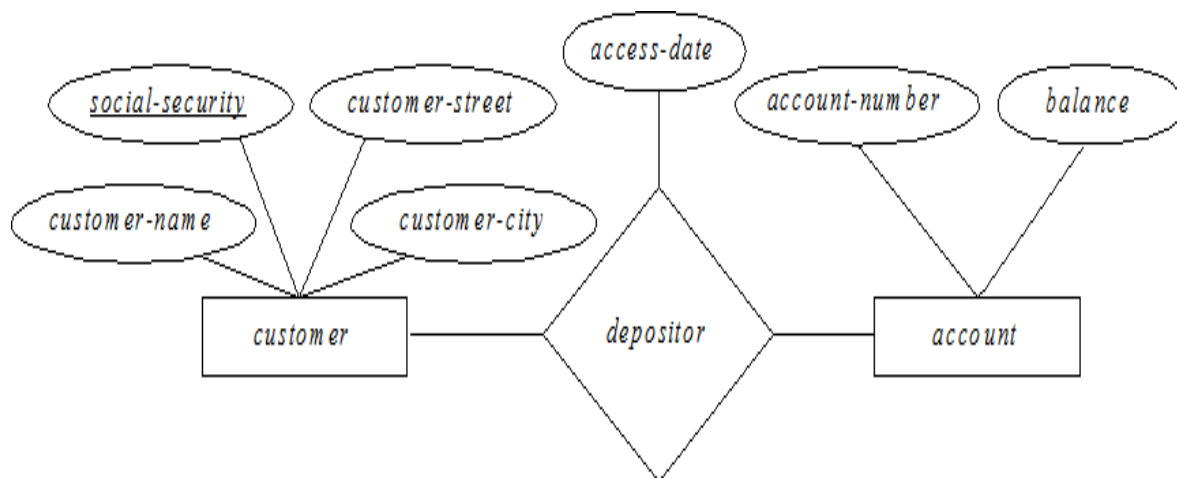
Relationship Sets

- A relationship is an association among several entities.
- A *relationship set* is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

where (e_1, e_2, \dots, e_n) is a relationship.

- An *attribute* can also be a property of a relationship set.
 - ❖ For instance, the *depositor* relationship set between entity sets.
 - ❖ *customer* and *account* may have the attribute *access-date*.



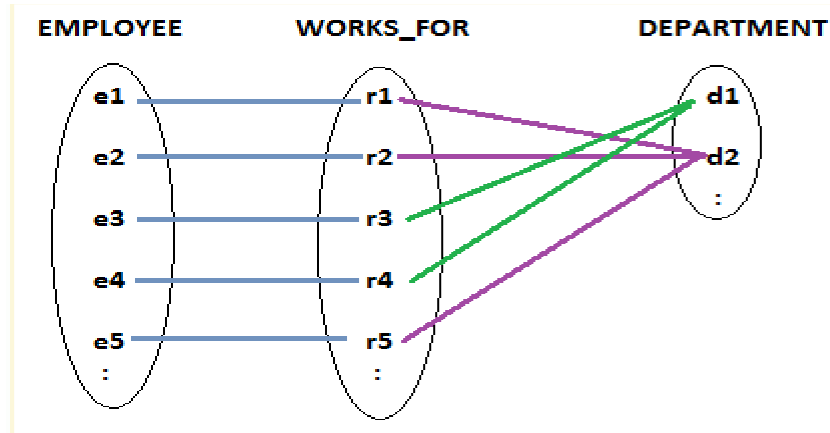
Degree of Relationship Sets

- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are binary (or degree two). Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets. The entity sets customer, loan, and branch may be linked by the ternary (degree three) relationship set CLB.

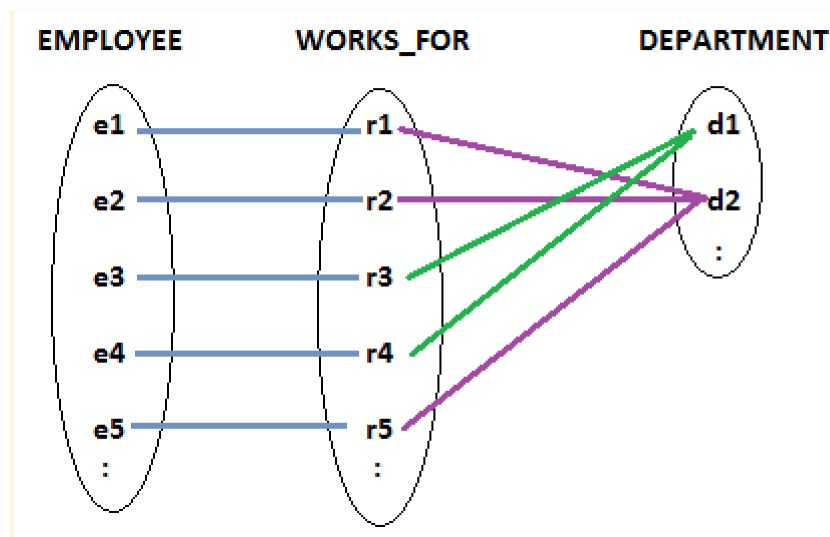
CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

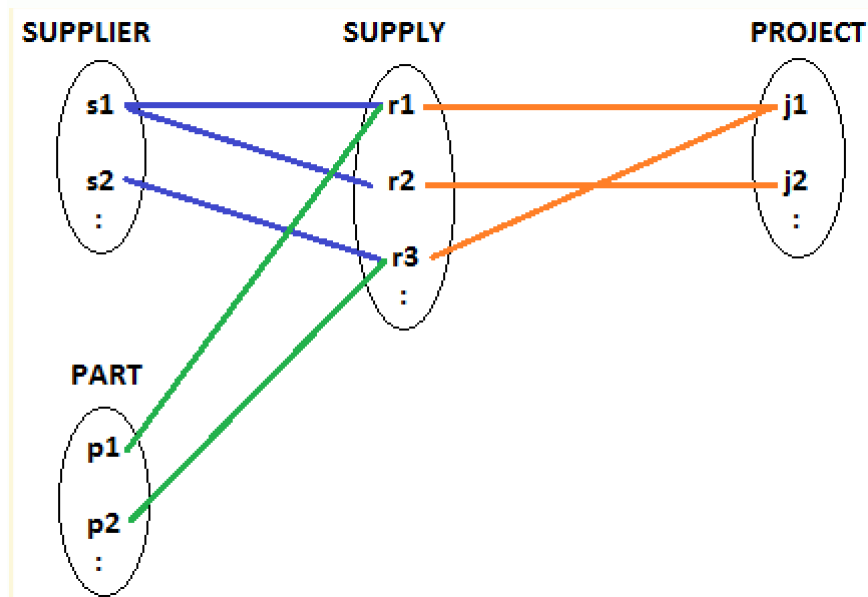
BATCH:2017-2020



- In the above figure, each instance of relation type WORKS_FOR i.e.(r1, r2,...,r5) is connected to instances of employee and department entities. Employee e1, e2 and e5 work for department d2 and employee e3 and e4 work for department d1.
- **Binary Relationship** – A relationship type of degree two is called binary relationship. The WORKS_FOR in above figure is a binary relationship as there are two participating entities-employee and department

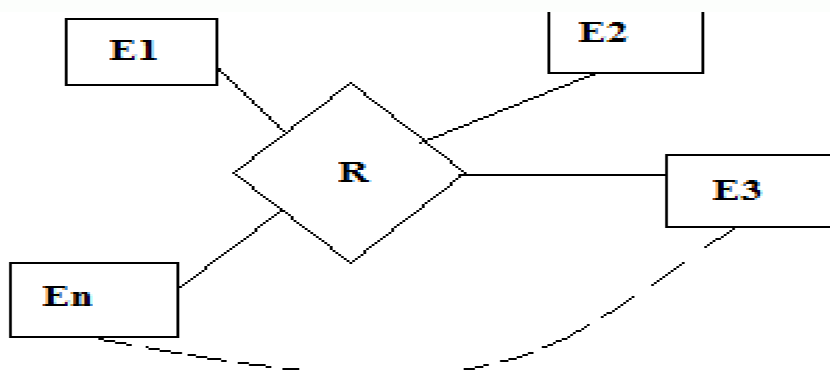


- **Ternary Relationship-** A relationship type of degree three is a ternary relationship for example, in the below figure **supply** relationship connects three entities SUPPLIER, PART AND PROJECT.



The above diagram can be read as – a supplier supplies the parts to projects

- **N-ary Relationship Set** – A relationship type of degree n is called n ary relationship . For example



Attributes of R --> Primary Key of all Entities

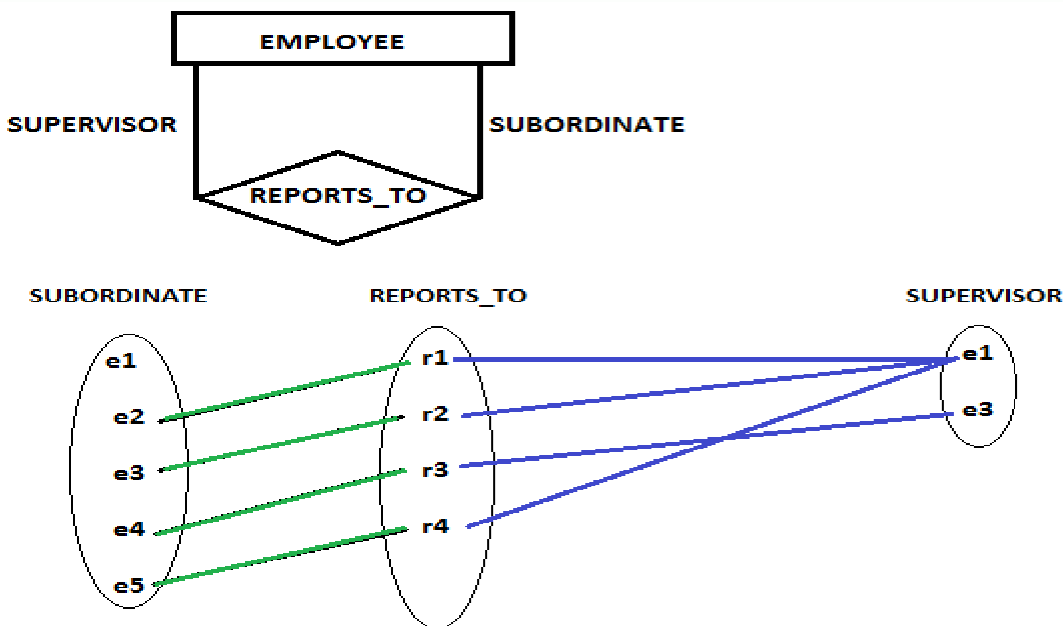
Role Names-

A relationship type has a name which signifies what role a participating entity plays in that relationship instance. The role names helps to explain what the relationship means.

In the first example WORKS_FOR relationship type, employee plays the role of worker and department plays the role of employee(because a department consists of a number of employees).

Recursive Relationship

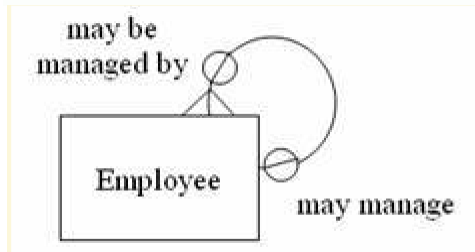
If the same entity type participate more than once in a relationship type in different roles then such relationship types are called recursive relationship. For example, in the below figure REPORTS_TO is a recursive relationship as the Employee entity type plays two roles – 1) Supervisor and 2) Subordinate.



CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)
BATCH:2017-2020

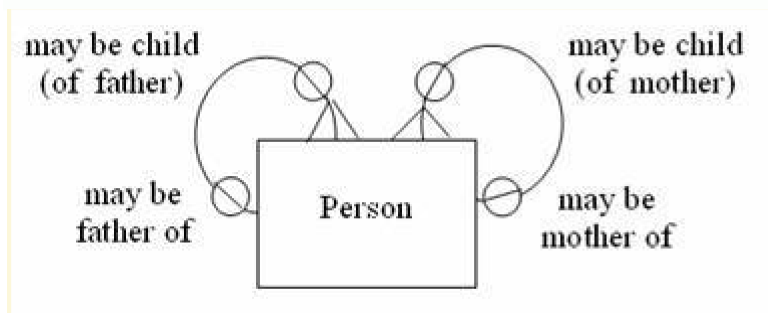
we can also define the above example of recursive relationship as the relationship between a manager and an employee. An employee is a manager as well as employee.



This is commonly known as a 'pig's ear'.

To implement recursive relationship, a foreign key of the employee's manager number would be held in each employee record.

Relationship between a person and their parents :



Mapping Cardinality-

Cardinality expresses the number of entities to which another entity can be associated via a relationship set (or) It specifies the number of relationship instances that an entity can participate in a relation set.

There are 4 types of Cardinality Ratios :

- **One-to-One Cardinality (1:1)**
- **One-to-Many Cardinality (1:m)**

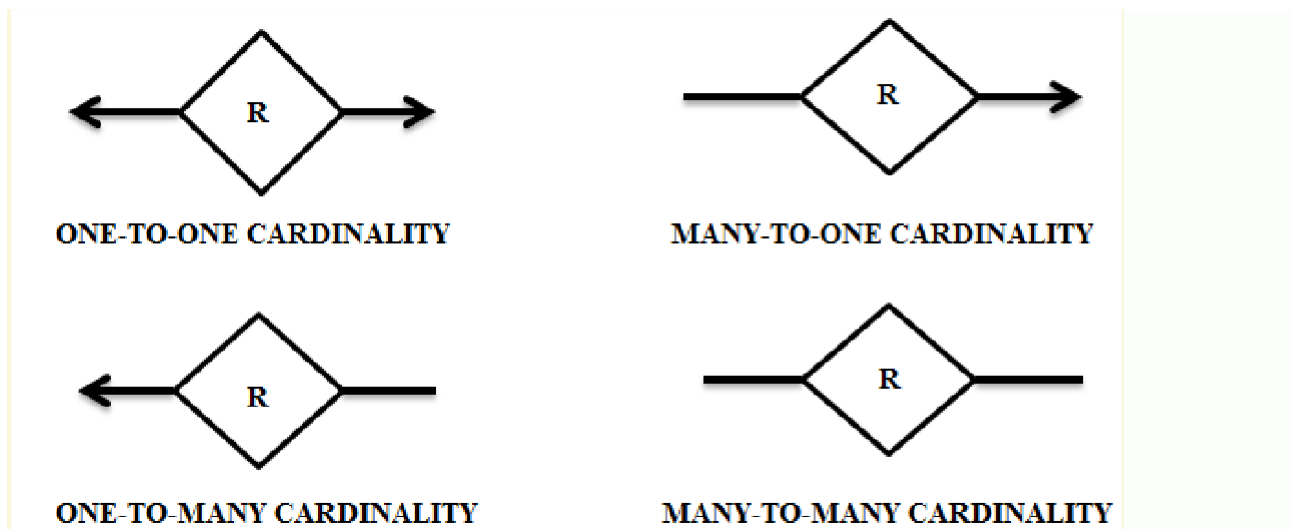
CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

- **Many-to-One Cardinality (m:1)**
- **Many-to-Many Cardinality (m:n)**

Notations of Different Types of Cardinality In ER Diagram –



One-to-One Cardinality (1:1) –

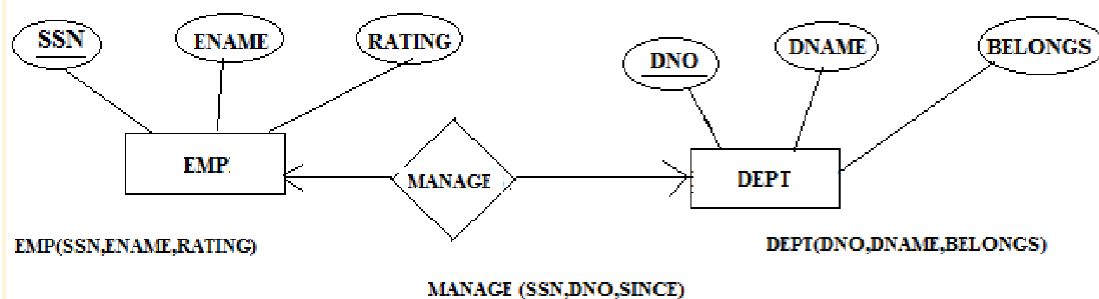
An entity in set A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A. This type of cardinality is referred to as one to one Cardinality.

For example, an Employee as a Manager manage only one Department and the vice versa is also true as a department have only one Manager.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020



Relational Tables -

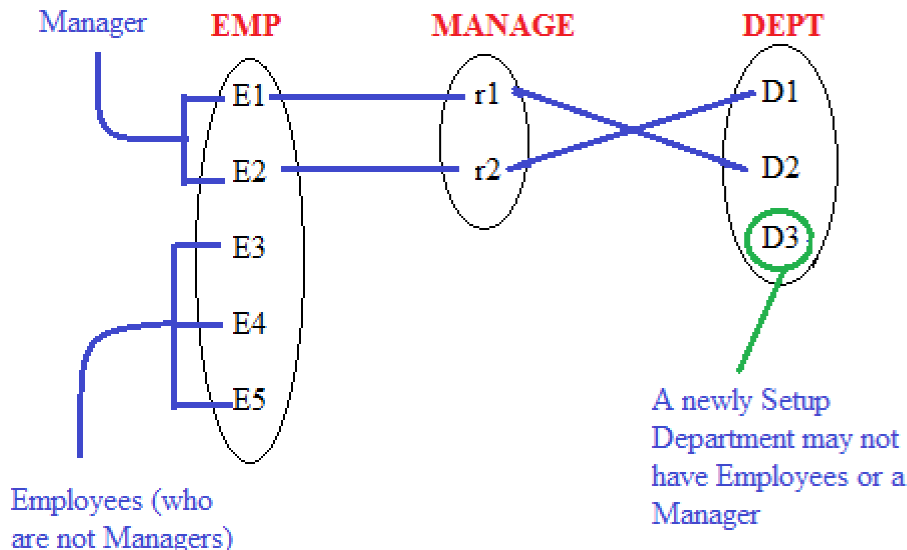
SSN	ENAME	RATING
E1		
E2		
E3		
E4		

SSN	DNO
E1	D2
E2	D1
E2	D2

DNO	DNAME	BELONGS
D1		
D2		
D3		
D4		
D5		

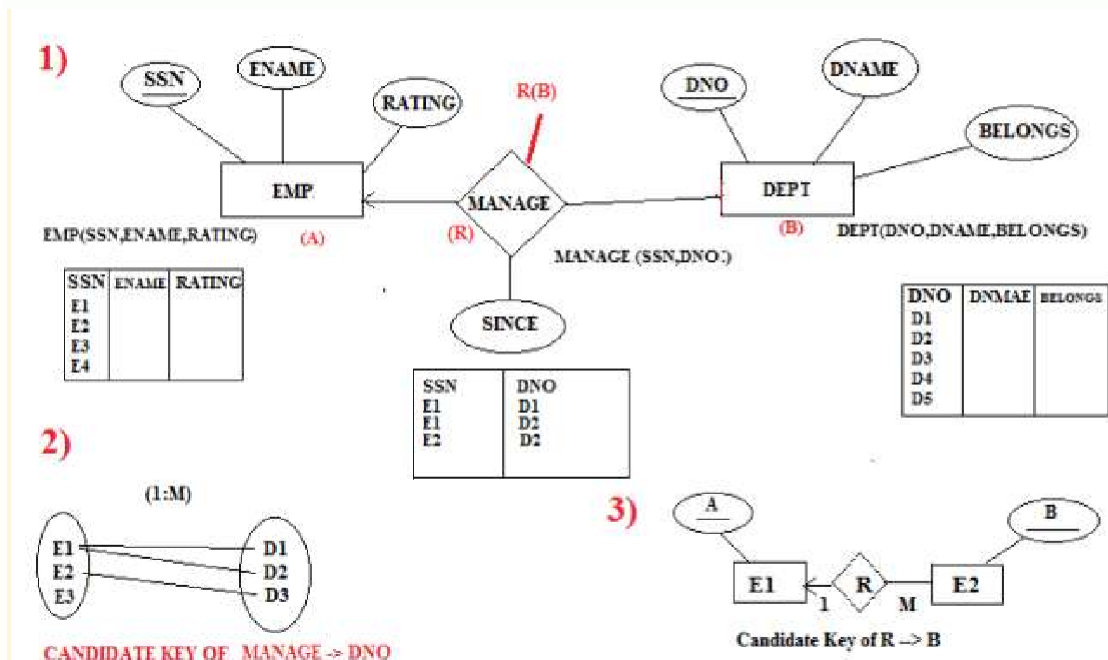
✗ (NOT POSSIBLE)
 AS A MANAGER CAN
 MANAGE ONLY ONE
 DEPARTMENT

Employees as a Manager



One to Many Cardinality (1:M) –

An entity in A is associated with any number (0 or more) with an entity B, but a entity in B, however can be associated with atmost one entity in A. For example, An employee as a Manager can manage more than one Department.



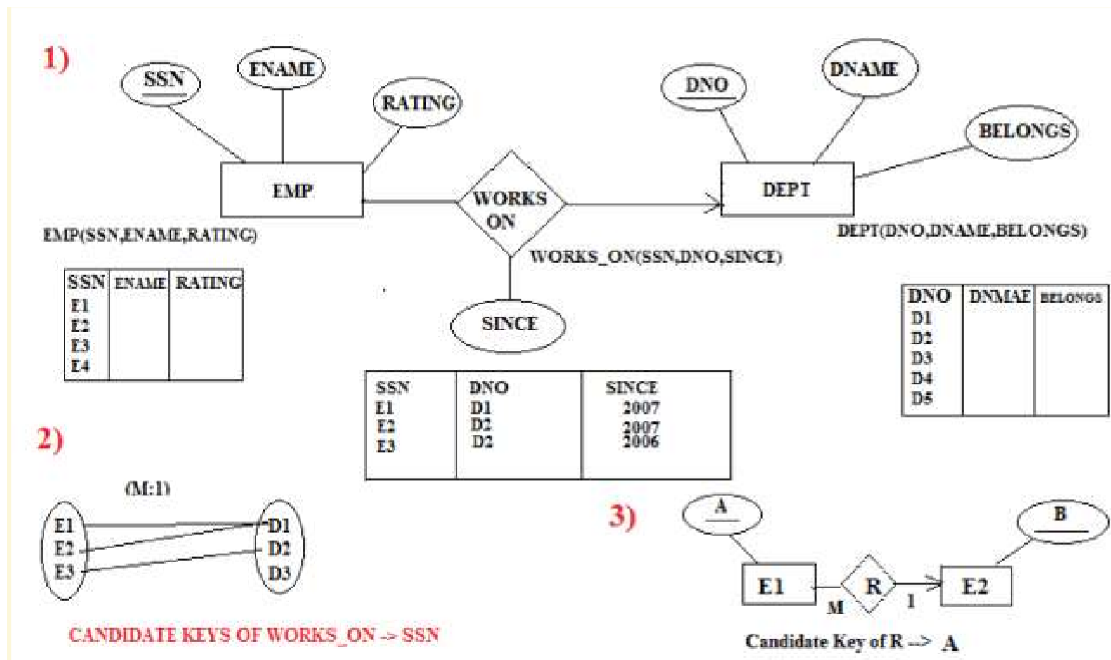
Many to One Cardinality (M:1) –

- An entity in A is associated with atmost one entity in B.
- An entity in B, however, can be associated with any number (0 or more) of entities in A.
- For example, An Employee can work only for one Department, But each Department can have 0 or more employees.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020



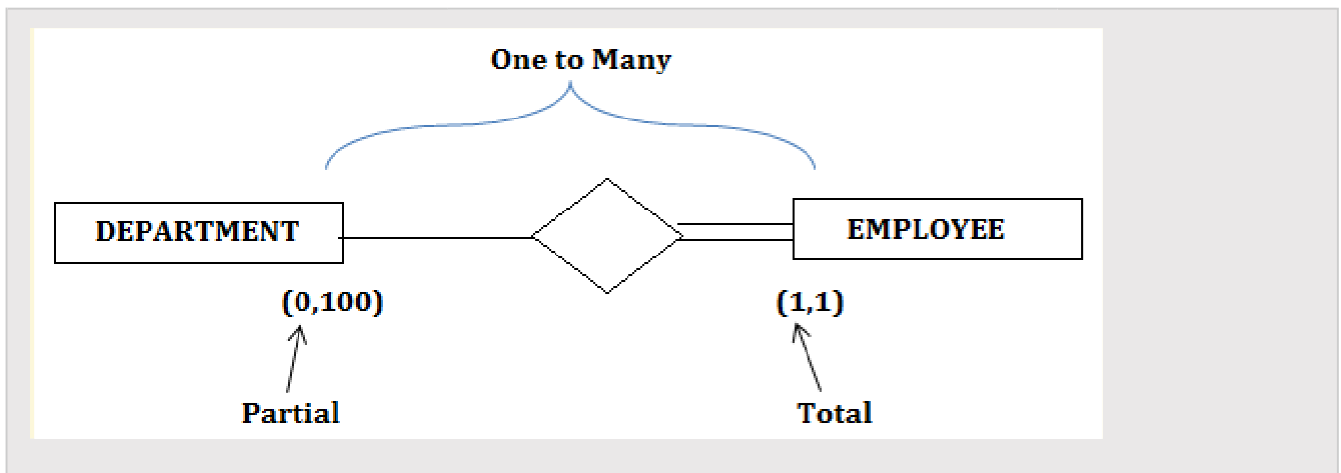
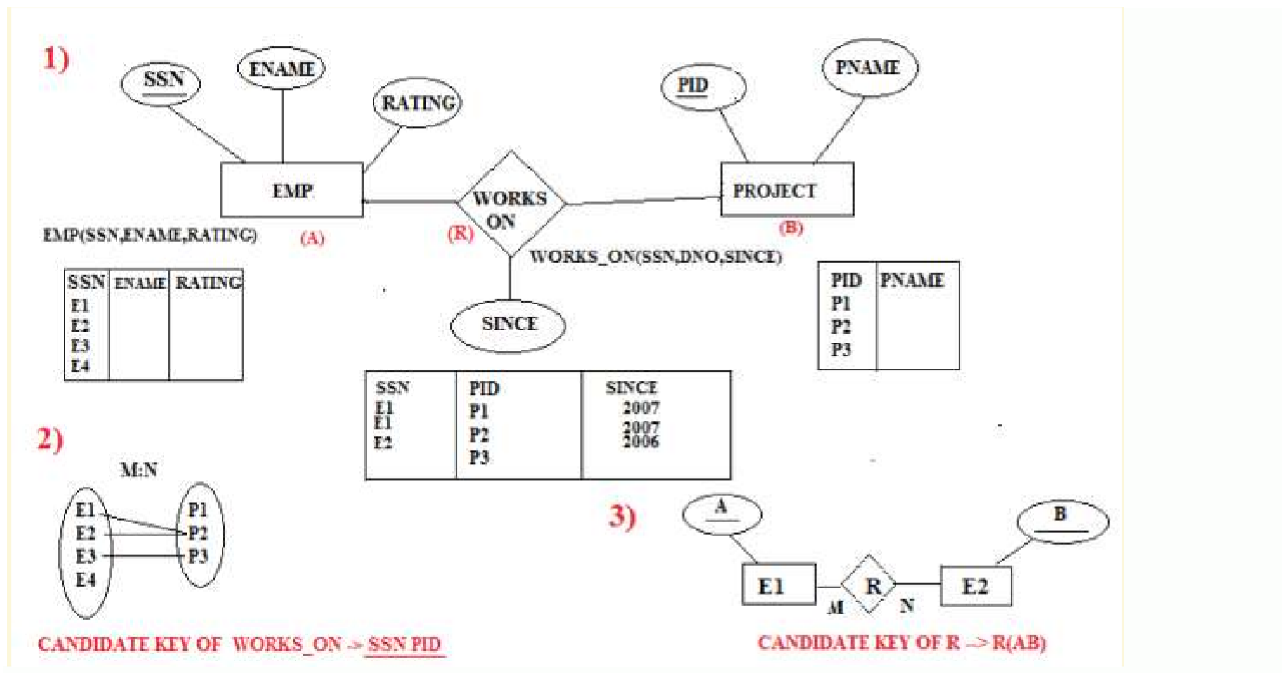
Many to Many Cardinality (M:N) –

- An entity in A is associated with any number (0 or more) of entities in B, and an entity in B is associated with any number (0 or more) of entities in A.
- For example, An Employee can work on several Projects and a Project may have several Employees.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020



E-R Diagrams in DBMS: Components, Symbols, and Notations

diagrams are used to model real-world objects like a person, a car, a company etc. and the relation between these real-world objects.

An e-r diagram has following features:

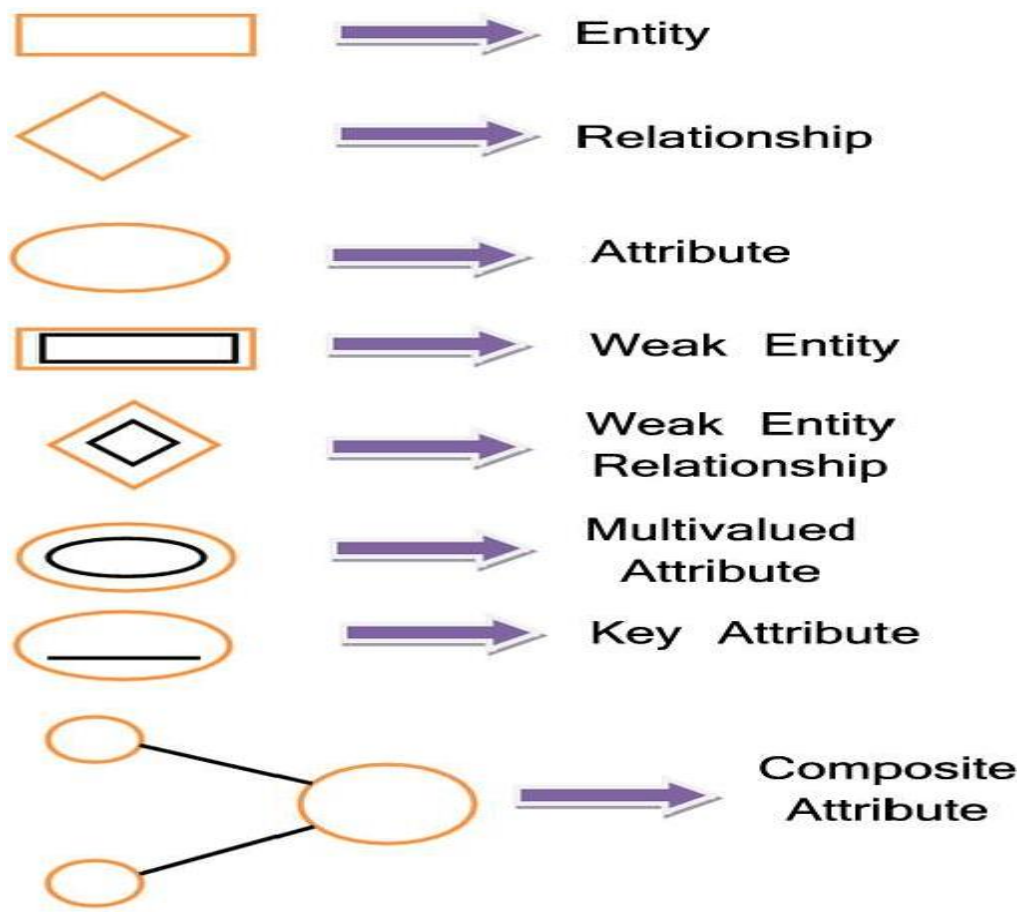
CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

- E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).
- E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- E-R diagrams require no technical knowledge & no hardware support.
- These diagrams are very easy to understand and easy to create even by a naive user.
- It gives a standard solution of visualizing the data logically.

ER Diagrams Symbols, And Notations



KEYS

- Key is a set of one or more columns whose combined values are *unique* among all occurrences in a given table.
- A key is the relationε l means of specifying uniqueness.

For example,

- In patient database, a patient number could be used as a key field to uniquely identify each patient's record in the doctors' patient file.
- The patient's name could not be a key field as there may be more than one patient with the same name. There must be at least one key field in each table.

TYPES OF KEYS

Primary Key

- A primary is a column or set of columns in a table that uniquely identifies tuples (rows) in that table.
- The candidate key which is very suitable to be the main key of table is a primary key.
- The primary keys are compulsory in every table.
- The properties of a primary key are:
 - ❖ Model stability
 - ❖ Occurrence of minimum fields
 - ❖ Defining value for every record i.e. being definitive
 - ❖ Feature of accessibility

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

Example

Student Id	First name of student	Last name of student	Course Id
123456	Jasmine	Shaik	001
123457	Rose	Mary	002
123458	Lily	Holmes	003

↓
Primary key

Super Key

- A super key is a set of one or more columns (attributes) to uniquely identify rows in a table.
- Super Key is a set of properties within a table; it specially identifies each record in a table. Candidate key is a unique case of super key.
- ❖ For example: Roll No. of a student is unique in relation. The set of properties like roll no., name, class, age, sex, is a super key for the relation student.

Candidate Key

- A candidate is a subset of a super key.
- A candidate key is a single field or the least combination of fields that uniquely identifies each record in the table.
- The least combination of fields distinguishes a candidate key from a super key.

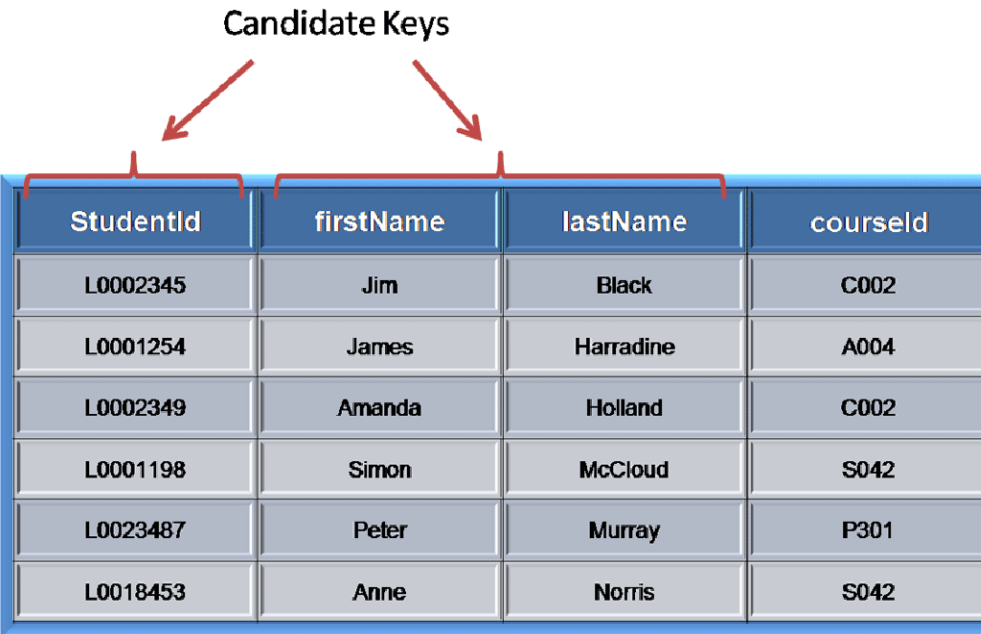
CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

- Every table must have at least one candidate key but at the same time can have several.

Candidate Keys



StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

Foreign Key

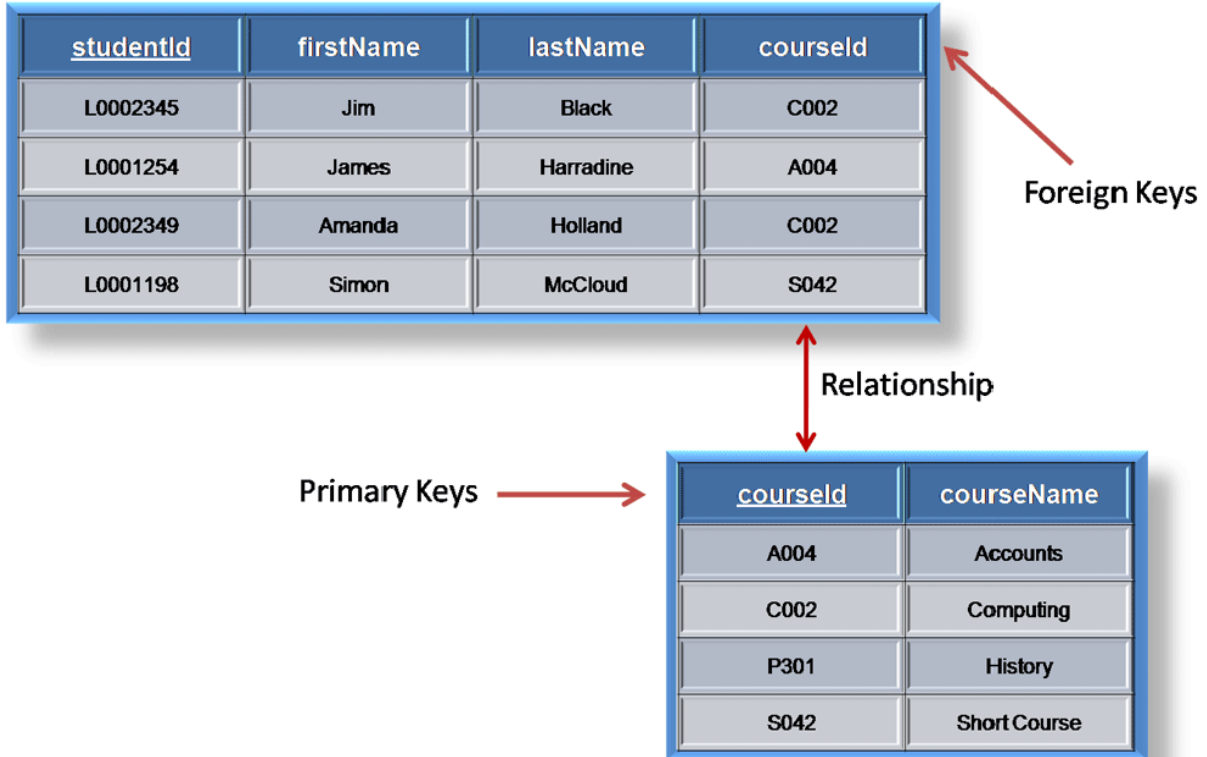
- A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second.
- In other words, if we had a table A with a primary key X that linked to a table B where X was a field in B, then X would be a foreign key in B.

For example, student table that contains the course_id the student is attending. Another table lists the courses on offer with course_id being the primary key. The 2 tables are linked through course_id and as such course_id would be a foreign key in the student table.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020



Secondary or Alternative key

- The rejected candidate keys as primary keys are called as secondary or alternative keys.

For Example:

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

Student Id	First name of student	Last name of student	Course Id
123456	Jasmine	Shaik	001
123457	Rose	Mary	002
123458	Lily	Holmes	003

Secondary or Alternative keys

Composite Key

- Key that consists of two or more attributes that uniquely identify an entity occurrence is called **Composite key**.
- But any attribute that makes up the **Composite key** is not a simple key in its own.

Composite Key

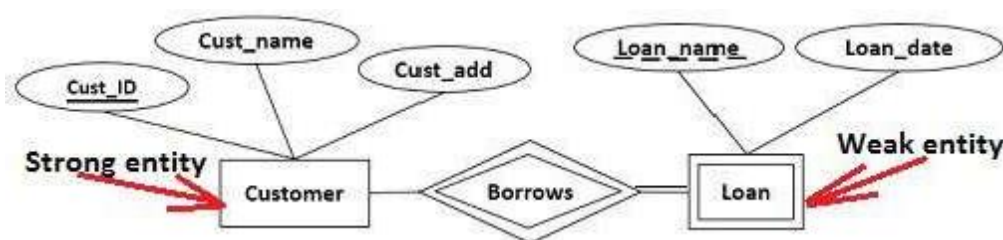


cust_id	order_id	sale_detail

Strong Entity Sets

- The **Strong Entity** is the one whose existence does not depend on the existence of any other entity in a schema.
- It is denoted by a **single rectangle**.
- A strong entity always has the **primary key** in the set of attributes that describes the strong entity.
- It indicates that each entity in a strong entity set can be uniquely identified.
- Set of similar types of strong entities together forms the **Strong Entity Set**.
- A strong entity holds the relationship with the weak entity via an **Identifying Relationship**, which is denoted by double diamond in the ER diagram.
- On the other hands, the relationship between two strong entities is denoted by a single diamond and it is simply called as a **relationship**.

For example; a customer borrows a loan. Here we have two entities first a customer entity, and second a loan entity.



Weak Entity Sets

- A **Weak entity** is the one that depends on its owner entity i.e. a strong entity for its existence.
- A weak entity is denoted by the **double rectangle**.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: I (Purpose of Database)

BATCH:2017-2020

- Weak entity do **not** have the **primary key** instead it has a **partial key** that uniquely discriminates the weak entities.
- The **primary key of a weak entity** is a composite key formed from the **primary key of the strong entity** and **partial key of the weak entity**.
- The collection of similar weak entities is called **Weak Entity Set**.
- The relationship between a weak entity and a strong entity is always denoted with an **Identifying Relationship** i.e. **double diamond**.

BASIS FOR COMPARISON	STRONG ENTITY	WEAK ENTITY
Basic	The Strong entity has a primary key.	The weak entity has a partial discriminator key.
Depends	The Strong entity is independent of any other entity in a schema.	Weak entity depends on the strong entity for its existence.
Denoted	Strong entity is denoted by a single rectangle.	Weak entity is denoted with the double rectangle.
Relation	The relation between two strong entities is denoted by a single diamond simply called relationship.	The relationship between a weak and a strong entity is denoted by Identifying Relationship denoted with double diamond.

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: I (Purpose of Database)

BATCH:2017-2020

BASIS FOR COMPARISON	STRONG ENTITY	WEAK ENTITY
Participation	Strong entity may or may not have total participation in the relationship.	Weak entity always has total participation in the identifying relationship shown by double line.



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-641021
(For candidates admitted from 2017 onwards)

SUBJECT NAME: DATABASE MANAGEMENT SYSTEM

SEMESTER: IV

SUBJECT CODE:17CCU402

UNIT-I

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	The data stored in the system is partitioned into one or more _____	Database	Data record	Data field	Stored field	Database
2	The database system involves _____ major components	2	4	5	3	4
3	The data is stored in _____ and _____	Concept & Physical	Shared & Concept	Integrated & Shared	Stored	Integrated & Shared
4	An _____ data refers to information entering the system from the outside world.	Output	Input	Process	Information	Input
5	An _____ data refers to messages and reports.	Output	Input	Process	Both input and output	Output

6	The _____ consists of secondary storage volumes.	Software	Hardware	Firmware	System	Hardware
7	The users of the system are a layer of _____ usually called the DBMS.	Software	Hardware	Firmware	Users	Software
8	A _____ is the application programmer .	Data	System	User	Information	User
9	A _____ is a collection of stored operational data.	Field	Database	Memory	Record	Database
10	The _____ linking the basic entities together.	Relationship	Sequence	Selection	Projection	Relationship

11	The _____ is impossible to change the storage structure	Data-independent	Data-dependent	Data-logic	Dependent	Data-dependent
12	The different application will need a different view of the same data called _____	Data-independent	Data-dependent	Data-logic	Independent	Data-dependent
13	A _____ must have the freedom to change the storage structure.	DBA	DML	DDA	TCL	DBA
14	A stored _____ is the smallest collection of associated stored fields.	Record	Data	File	Field	Record
15	A stored _____ is the smallest named unit of data stored in the database.	Record	Data	File	Field	Field
16	A stored _____ is the collection of occurrences of one type of stored record.	Record	Data	File	Field	File
17	A _____ field may be stored in internal arithmetic form or a character string.	Char	Integer	Numeric	Float	Numeric
18	A _____ string field may be stored in any of several character codes.	Char	Integer	Numeric	Float	Char

19	The _____ in a numeric field may change from inches to centimeters.	Char	Integer	Numeric	Units	Units
20	The _____ may be desirable to represent data in storage by coded values.	Data coding	Char	Integer	Data byte	Data coding
21	The _____ constructing an occurrence of the logical field from the corresponding stored field occurrence and presenting it to the application.	Data materialization	Data independence	Data dependence	Record	Data materialization
22	A _____ file may be physically implemented in storage in a wide variety of ways.	Accessed	Related	Stored	Record	Stored
23	The architecture is divided into — levels.	4	2	5	3	3

24	The____level is the one closest to physical storage.	Internal	External	Logical	Conceptual	Internal
25	The_____ level is the one closest to the user.	Internal	External	Logical	Conceptual	External
26	The____level is a level of indirection.	Internal	External	Logical	Conceptual	Conceptual
27	The_____ is a subset of the total language that is concerned with database objects and operations.	DSL	DML	DDL	DCL	DSL

28	The data sublanguage as being embedded in a _____ language.	Source	Host	Designation	Destination	Host
29	The _____ provides for the definition or description of database objects	DSL	DML	DDL	TCL	DDL
30	A _____ supports the manipulation or processing of such objects	DSL	DML	DDL	TCL	DML
31	A _____ record is not necessarily the same as a stored record.	Internal	External	Logical	Conceptual	External
32	A _____ record is not necessarily the same as either an external record, on the one hand, or a stored record on the other.	Internal	External	Logical	Conceptual	Conceptual
33	The _____ database that is not stored in its entirety at a single physical location.	Shared	Source	Distributed	Designation	Distributed
34	The component responsible for this internal /physical conversion is called a _____	Access method	Record	Field method	Interface	Access method

35	A _____ key values uniquely identify the records in the file.	Foreign	Candidate	Primary	Superkey	Primary
36	The _____ is not stored in its entity at a single location but rather is spread across a network of computers.	Network Database	Distributed Database	Database	Central database	Distributed Database
37	The physical record is the interface between the _____ method and the _____ database.	Access & Internal	Access & Physical	Access & External	Access & Conceptual.	Access & Physical
38	The indexed file can be accessed through direct and _____ access	Indirect	Sequential access	Direct & Indirect	Direct	Indirect
39	The _____ set provides fast direct access to the data	Sequence	Group	Index	Record	Index
40	The conceptual schema can sometimes be called as _____	Logical schema	Physical schema	External schema	Internal schema	Logical schema
41	An each stored file is sequenced by the access method on its _____	Primary Key	Foreign Key	Candidate Key	Super key	Primary Key
42	The purpose of index is to provide an _____ path to the file.	Access	Root	Unaccessed	Direct	Access

43	The _____ symbol is used to mean pointer.	Up arrow	Down arrow	Middle arrow	Line	Up arrow
44	The rows of such tables are generally referred to as _____	Tuples	Attributes	Domain	Index	Tuples
45	The columns of such tables are generally referred to as _____	Tuples	Attributes	Domain	Index	Attributes
46	A _____ is a pool of _____ values from which the actual values appearing in a given column are drawn.	Tuples	Attributes	Domain	Index	Domain
47	The _____ view data is represented by a simple tree structure.	Relational	Hierarchical	Network	Non-relational	Hierarchical
48	A record type at the top of the tree is usually known as _____	Root	Child	Index	Record	Root
49	A _____ connecting occurrences of records.	Root node	Link	File	Child node	Link
50	The _____ is a major drawback of the hierarchical approach	Symmetry	Asymmetric	Reference	Relational data	Asymmetric
51	The _____ to change the description of a supplier.	Insert	Delete	Update	View	Update

52	In the _____ approach the data is represented by records and links.	Relational	Hierarchical	Network	stored interface	Network
53	The _____ approach allow us to model a many to many correspondences more directly.	Relational	Hierarchical	Network	Field	Network
54	The _____ associations between tuples are represented solely by data values in columns drawn from a common domain.	Relational data structure	Domain	Keys	Record	Relational data structure
55	The _____ is a term that has a much more precise definition than the more traditional data processing.	Relational	Hierarchical	Network	Rows	Relational
56	The relational model is divided into _____ parts	3	2	5	1	3
57	The _____ is mostly calling as a table	Tuple	Degree	Primary key	Relation	Relation
58	An _____ correspondends to the row of a table	Attribute	Cardinality	Primary key	Tuple	Attribute
59	An _____ is a unique definer of a table	Domain	Cardinality	Primary key	Tuple	Domain
60	A _____ is a value from which specific attributes are drawn their actual values	Domain	Relation	Degree	Tuple	Degree

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: II (Relational Model) **BATCH: 2017-2020**

UNIT II

SYLLABUS

Relational Model - Structure - Formal Query Language - Relational Algebra - Tuple and Domain Relational Calculus.

Relational Model-Introduction

Relational model stores data in the form of tables. This concept purposed by Dr. E.F. Codd, a researcher of IBM in the year 1960s. The relational model consists of three major components:

1. The set of relations and set of domains that defines the way data can be represented (data structure).
2. Integrity rules that define the procedure to protect the data (data integrity).
3. The operations that can be performed on data (data manipulation).

A relational model database is defined as a database that allows you to group its data items into one or more independent tables that can be related to one another by using fields common to each related table.

Characteristics of Relational Database

Relational database systems have the following characteristics:

- The whole data is conceptually represented as an orderly arrangement of data into rows and columns, called a relation or table.
- All values are scalar. That is, at any given row/column position in the relation there is one and only one value.

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: II (Relational Model)

BATCH: 2017-2020

All operations are performed on an entire relation and result is an entire relation, a concept known as closure.

Dr. Codd, when formulating the relational model, chose the term "relation" because it was comparatively free of connotations, unlike, for example, the word "table". It is a common misconception that the relational model is so called because relationships are established between tables. In fact, the name is derived from the relations on whom it is based. Notice that the model requires only that data be conceptually represented as a relation, it does not specify how the data should be physically implemented. A relation is a relation provided that it is arranged in row and column format and its values are scalar. Its existence is completely independent of any physical representation.

Structure of Relational Database

1. A relational database consists of a collection of **tables**, each having a unique **name**.

A **row** in a table represents a **relationship** among a set of values.

Thus a table represents a **collection of relationships**.

2. There is a direct correspondence between the concept of a table and the mathematical concept of a relation. A substantial theory has been developed for relational databases.

1. Figure shows the *deposit* and *customer* tables for our banking example.

Bname	Account	Ename	Balance
Downtown	101	Johnson	300
Longheed Mall	215	Smith	700
SPU	102	Agnes	400
SPU	204	Adams	1300

Ename	Street	City
Johnson	Pender	Vancouver
Smith	North	Burnaby
Agnes	Cintia	Burnaby
Adams	Nod Road	Richmond
Jones	Oak	Vancouver

The *deposit* and *customer* relations.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: II (Relational Model) **BATCH: 2017-2020**

- It has four attributes.
- For each attribute there is a permitted set of values, called the **domain** of that attribute.
- E.g. the domain of *bname* is the set of all branch names.

Let D_1 denote the domain of *bname*, and D_2, D_3, D_4 the remaining attributes' the domains respectively.

Then, any row of *deposit* consists of a four-tuple (v_1, v_2, v_3, v_4) where $v_1 \in D_1, v_2 \in D_2, v_3 \in D_3, v_4 \in D_4$

In general, *deposit* contains a **subset** of the set of all possible rows.

That is, *deposit* is a subset of

$$D_1 \times D_2 \times D_3 \times D_4 \text{ or abbreviated to } \prod_{i=1}^4 D_i$$

In general, a table of n columns must be a subset of

$$\prod_{i=1}^n D_i$$

2. Mathematicians define a relation to be a subset of a Cartesian product of a list of domains.
3. The terms **relation** and **tuples** in place of **table** and **row**.
4. Some more formalities:
 - let the tuple variable **t** refer to a tuple of the relation **r**.
 - We say $t \in r$ to denote that the tuple **t** is in relation **r**.
 - Then $t[bname] = t[1] =$ the value of **t** on the *bname* attribute.
 - So $t[bname] = t[1] = \text{"Downtown"}$,

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

- and $t[cname] = t[3] = \text{"Johnson"}$.

5. We'll also require that the domains of all attributes be **indivisible units**.

- A domain is **atomic** if its elements are indivisible units.
- For example, the set of integers is an atomic domain.
- The set of all sets of integers is not.
- Integers do not have subparts, but sets do - the integers comprising them.
- We could consider integers non-atomic if we thought of them as ordered lists of digits.

Keys

1. The notions of **superkey**, **candidate key** and **primary key** all apply to the relational model.
2. For example, in *Branch-scheme*,
 - $\{bname\}$ is a superkey.
 - $\{bname, bcity\}$ is a superkey.
 - $\{bname, bcity\}$ is not a candidate key, as the superkey $\{bname\}$ is contained in it.
 - $\{bname\}$ is a candidate key.
 - $\{bcity\}$ is not a superkey, as branches may be in the same city.
 - We will use $\{bname\}$ as our primary key.
3. The primary key for *Customer-scheme* is $\{cname\}$.
4. More formally, if we say that a subset K of R is a *superkey* for R , we are restricting consideration to relations $r(R)$ in which no two distinct tuples have the same values on all attributes in K . In other words,
 - If t_1 and t_2 are in R and $t_1 \neq t_2$, then $t_1(K) \neq t_2(K)$

Query Languages

1. A **query language** is a language in which a user requests information from a database. These are typically higher-level than programming languages.

They may be one of:

- **Procedural**, where the user instructs the system to perform a sequence of operations on the database. This will compute the desired information.
 - **Nonprocedural**, where the user specifies the information desired without giving a procedure for obtaining the information.
2. A complete query language also contains facilities to insert and delete tuples as well as to modify parts of existing tuples.

The Relational Algebra

1. The **relational algebra** is a procedural query language.
 - Six fundamental operations:
 - select (unary)
 - project (unary)
 - rename (unary)
 - cartesian product (binary)
 - union (binary)
 - set-difference (binary)
 - Several other operations, defined in terms of the fundamental operations:
 - set-intersection
 - natural join

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

- division
- assignment
- Operations produce a new relation as a result.

Student(sid, name, addr, age, GPA)

Sid	Name	Addr	Age	GPA
301	John	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Jaci	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

Dept	Cnum	Sec	Unit	Title	Instructor
CS 112	01	03	Modeling	Dick	Muntz
CS 143	01	04	DB Systems	Carlo	Zaniolo
EE 143	01	03	Signal	Dick	Muntz
ME 183	02	05	Mechanics	Susan	Tracey

Enroll(sid, dept, cnum, sec)

Sid	Dept	Cnum	Sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Select operation(σ): To identify a set of tuples which is a part of a relation and to extract only these tuples out. The select operation selects tuples that satisfy a given predicate or condition.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

- It is a unary operation defined on a single relation.
- It is denoted as σ .

This operation pulls the horizontal subset (subset of rows) of the relation that satisfies the conditions. This can use operators like $<$, $>$, \leq , \geq , $=$ and \neq to filter the data from the relation. It can also use logical AND, OR and NOT operators to combine the various filtering conditions. This operation can be represented as below:

$\sigma P (r)$

Where σ is the symbol for select operation, r represents the relation/table, and p is the logical formula or the filtering conditions to get the subset.

Select all tuples satisfying a condition

- Query: Students with age < 18

$\sigma_{\text{age} < 18} (\text{Student})$

Sid	Name	Addr	Age	GPA
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5

- Query: Students with GPA > 3.7 and age < 18

$\sigma_{\text{age} < 18 \wedge \text{GPA} > 3.7} (\text{Student})$

Sid	Name	Addr	Age	GPA
303	Elaine	301 Wilshire	17	3.9

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: II (Relational Model)
BATCH: 2017-2020

Project operation(π)

$\pi_{A_1, A_2, \dots, A_k}(r)$

where A_1, A_2 are attribute names and r is a relation name.

The result is defined as the relation of k columns obtained by erasing the columns that are not listed. Duplicate rows removed from result, since relations are sets

This is a unary operator and is similar to select operation above. It creates the subset of relation based on the conditions specified. Here, it selects only selected columns/attributes from the relation- vertical subset of relation. The select operation above creates subset of relation but for all the attributes in the relation.

It is a unary operation defined on a single relation

Query: sid and GPA of all students

$\pi_{sid, GPA}(Students)$

Sid	Name
301	John
303	Elaine
401	James
208	Jaci

Query: All departments offering classes

$\pi_{dept}(Class)$

Dept
CS 112
CS 143
EE 143
ME 183

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

Composition of Relational Operations

A relational – algebra operation is of the same type (relation) as its inputs, relational-algebra operations can be composed together into a relational-algebra expression.

Query: Find that student who has enrolled in dept “cs”.

$\Pi_{sid} (\sigma_{dept = \text{“cs”}} (\text{Enroll}))$

Sid
301
303
401

Union Operation (U)

Defined as:

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

It is a binary operator, which combines the tuples of two relations. It is denoted by

R U S

Where R and S are the relations and U is the operator.

It is different from cartesian product in:

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

Cartesian product combines the attributes of two relations into one relation whereas Union combines the tuples of two relations into one relation.

In Union, both relations should have same number of columns. Suppose we have to list the employees who are working for design and testing department. Then we will do the union on employee table. Since it is union on same table it has same number of attributes. Cartesian product does not concentrate on number of attribute or rows. It blindly combines the attributes.

In Union, both relations should have same types of attributes in same order. In the above example, since union is on employee relation, it has same type of attribute in the same order.

It need not have same number of tuples in both the relation. If there is a duplicate tuples as a result of union, then it keeps only one tuple. If a tuple is present in any one relation, then it keeps that tuple in the new relation. In the above example, number of employees in design department need not be same as employees in testing department. Below diagram shows the same. We can observe that it combines the table data in the order they appear in the table.

Design_Employee

Emp-id	EName
100	James
104	Kathy

U

Testing_Employee

Emp-id	EName
103	Rose
102	Mary
105	Laury

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

Emp- id	ENAME
100	James
104	Kathy
103	Rose
102	Mary
105	Laury

Cartesian product (X): - This is a binary operator. It combines the tuples of two relations into one relation.

Example: $R \times S$

A		B		A	B
a1	X	b1	=	a1	b1
a2		b2		a1	b2
		b3		a1	b3
				a2	b1
				a2	b2
				a2	b3

$R \times S$

Where R and S are two relations and X is the operator. If relation R has m tuples and relation S has n tuples, then the resultant relation will have mn tuples. For example, if we perform cartesian product on EMPLOYEE (5 tuples) and DEPT relations (3 tuples), then we will have new tuple with 15 tuples.

For Example

EMPLOYEE X DEPT

This operator will simply create a pair between the tuples of each table. i.e.; each employee in the EMPLOYEE table will be mapped with each department in DEPT table.

Below diagram depicts the result of cartesian product

Employee

Emp-id	EName
100	James

Dept

Dept-id	Dept-Name
10	Account

X

101	Kathy
102	Joseph
103	Rose
104	Mary

20	Design
30	Testing

Emp-id	EName	Dept-id	Dept-Name
100	James	10	Account
100	James	20	Design
100	James	30	Testing
101	Kathy	10	Account
101	Kathy	20	Design
101	Kathy	30	Testing
102	Joseph	10	Account
102	Joseph	20	Design
102	Joseph	30	Testing
103	Rose	10	Account
103	Rose	20	Design
103	Rose	30	Testing
104	Mary	10	Account
104	Mary	20	Design
104	Mary	30	Testing

Rename (ρ) - This is a unary operator used to rename the tables and columns of a relation. When we perform self join operation, we have to differentiate two same tables. In such case rename operator on tables comes into picture. When we join two or more tables and if those tables have same column names, then it is always better to rename the columns to differentiate them. this occurs when we perform Cartesian product

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

$\rho_x(E)$

Where ρ is the rename operator, E is the existing relation name, and x is the new relation name.

$\rho_{EMP}(EMPLOYEE)$ – Renames EMPLOYEE table to EMP

If a relational-algebra expression E has arity n , then returns the result of expression E under the name X , and with the attributes renamed to A_1, A_2, \dots, A_n .

$\rho_x(A_1, A_2, \dots, A_n)$

Example

Find the largest salary in the university

– Step 1: find instructor salaries that are less than some other

instructor salary (i.e. not maximum)

–using a copy of instructor under a new name d

$\Pi_{\text{instructor.salary}} (\sigma_{\text{instructor.salary} < d.\text{salary}}(\text{instructor} \times \rho_d(\text{instructor})))$

– Step 2: Find the largest salary

$\Pi_{\text{salary}}(\text{instructor}) - \Pi_{\text{instructor.salary}} (\sigma_{\text{instructor.salary} < d.\text{salary}}(\text{instructor} \times \rho_d(\text{instructor})))$

Set Difference(-)

Defined as:

$r - s = \{t \mid t \in r \text{ and } t \notin s\}$

For $R - S$ The Set difference operation defines a relation consisting of the tuples that are in relation

R , but not in S . R and S must be union-compatible.

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

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

Set differences must be taken between compatible relations.

- r and s must have the same arity
- attribute domains of r and s must be compatible

Formal Definition

A basic expression in the relational algebra consists of either one of the following:

- A relation in the database
- A constant relation
- ❖ Let E1 and E2 be relational-algebra expressions; the following are all relational-algebra expressions:

– $E1 \cup E2$

– $E1 - E2$

– $E1 \times E2$

– $\sigma_p(E1)$, P is a predicate on attributes in E1

– $\pi_s(E1)$, S is a list consisting of some of the attributes in E1

– $\rho_x(E1)$, x is the new name for the result of E1

Additional Operations

Additional operations are defined in terms of the fundamental operations. They do not add power to the algebra, but are useful to simplify common queries.

1. The Set Intersection Operation

Set intersection is denoted by \cap , and returns a relation that contains tuples that are in **both** of its argument relations.

It does not add any power as

$$r \cap s = r - (r - s)$$

To find all customers having both a loan and an account at the SFU branch, we write

$$\Pi_{\text{ename}} (\sigma_{\text{bname} = \text{"SFU"}} (\text{borrow})) \cap \Pi_{\text{ename}} (\sigma_{\text{bname} = \text{"SFU"}} (\text{deposit}))$$

2. The Natural Join Operation

Often we want to simplify queries on a cartesian product.

For example, to find all customers having a loan at the bank and the cities in which they live, we need *borrow* and *customer* relations:

$$\Pi_{\text{borrow.ename,city}} (\sigma_{\text{borrow.ename} = \text{customer.ename}} (\text{borrow} \times \text{customer}))$$

Our selection predicate obtains only those tuples pertaining to only one *cname*.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

This type of operation is very common, so we have the **natural join**, denoted by a \bowtie sign. Natural join combines a cartesian product and a selection into one operation. It performs a selection forcing equality on those attributes that appear in both relation schemes. Duplicates are removed as in all relation operations.

To illustrate, we can rewrite the previous query as

$\Pi_{\text{ename,city}} (\text{borrow} \bowtie \text{customer})$

Ename	City
Smith	Burnaby
Hayes	Burnaby
Jones	Vancouver

Joining *borrow* and *customer* relations.

We can now make a more formal definition of natural join.

- Consider R and S to be **sets** of attributes.
- We denote attributes appearing in **both** relations by $R \cap S$.
- We denote attributes in **either or both** relations by $R \cup S$.
- Consider two relations $r(R)$ and $s(S)$.
- The natural join of r and s , denoted by $r \bowtie s$ is a relation on scheme $R \cup S$.
- It is a projection onto $R \cup S$ of a selection on $r \times s$ where the predicate requires $r \wedge s$ for attribute $r \wedge R \cap S$. Formally

$$r \bowtie s = \Pi_{R \cup S} (\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \dots \wedge r.A_n = s.A_n} (r \times s))$$

where $R \cap S = \{A_1, A_2, \dots, A_n\}$.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

To find the assets and names of all branches which have depositors living in Stamford, we need *customer*, *deposit* and *branch* relations:

$$SELECT \text{branch_name, assets} FROM \text{branch, deposit, customer} WHERE \text{branch_name} = \text{branch_name} \text{ AND } \text{branch_name} = \text{branch_name}$$

Note that \cap is associative.

To find all customers who have both an account and a loan at the SFU branch:

This is equivalent to the set intersection version we wrote earlier. We see now that there can be several ways to write a query in the relational algebra.

If two relations R_1 and R_2 have no attributes in common, then $R_1 \cap R_2 = \emptyset$, and $R_1 \cup R_2 = R_1 \bowtie R_2$.

3. The Division Operation

Division, denoted \div , is suited to queries that include the phrase "for all".

Suppose we want to find all the customers who have an account at **all** branches located in Brooklyn.

Strategy: think of it as three steps.

We can obtain the names of all branches located in Brooklyn by

$$SELECT \text{branch_name} FROM \text{branch} WHERE \text{branch_name} = \text{branch_name}$$

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: II (Relational Model) BATCH: 2017-2020

We can also find all *cname*, *bname* pairs for which the customer has an account by

—.

Now we need to find all customers who appear in with **every** branch name in .

The divide operation provides exactly those customers:

which is simply .

Formally,

- Let R_1 and R_2 be relations.
- Let $R_1 \div R_2$.
- The relation $R_1 \div R_2$ is a relation on scheme S .
- A tuple t is in $R_1 \div R_2$ if for every tuple s in R_2 there is a tuple u in R_1 satisfying both of the following:

[—

- These conditions say that the i th portion of a tuple u is in R_1 if and only if there are tuples s with the i th portion **and** the j th portion in R_2 for **every** value of the j th portion in relation R_2 .

We will look at this explanation in class more closely.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

The division operation can be defined in terms of the fundamental operations.

— — — . —

Read the text for a more detailed explanation.

4. The Assignment Operation

Sometimes it is useful to be able to write a relational algebra expression in parts using a temporary relation variable (as we did with and in the division example).

The assignment operation, denoted , works like assignment in a programming language.

We could rewrite our division definition as

No extra relation is added to the database, but the relation variable created can be used in subsequent expressions. Assignment to a permanent relation would constitute a modification to the database.

Relational calculus

Relational calculus is an alternative to relational algebra. In contrast to the algebra, which is procedural, the relational calculus is non-procedural or declarative.

It uses mathematical predicate calculus. The relational calculus is not the same as that of differential and integral calculus in mathematics but takes its name from a branch of

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

symbolic logic termed as predicate calculus. When applied to databases, it is found in two forms.

These are

- Tuple relational calculus which was originally proposed by Codd in the year 1972 and
- Domain relational calculus which was proposed by Lacroix and Pirotte in the year 1977
- ❖ It is a formal language based upon a branch of mathematical logic called "predicate calculus".
- ❖ Variables in Tuple relational calculus takes tuples (rows) as values and it had strong influence on SQL.
- ❖ Variables in Domain relational calculus takes fields (attributes) as values and it had strong influence on QBE.

Tuple Relational Calculus

1. The tuple relational calculus is a nonprocedural language. (The relational algebra was procedural).

We must provide a formal description of the information desired.

2. A query in the tuple relational calculus is expressed as

$$\{t \mid P(t)\}$$

i.e. the set of tuples t for which predicate $P(t)$ is true.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

3. We also use the notation

- $t[a]$ to indicate the value of tuple t on attribute a .
- $t \in r$ to show that tuple t is in relation r .

Example Queries

1. For example, to find the branch-name, loan number, customer name and amount for loans over \$1200:

$$\{t \mid t \in r \text{ borrower} \wedge t[\text{amount}] > 1200\}$$

This gives us all attributes, but suppose we only want the customer names. (We would use **project** in the algebra.)

We need to write an expression for a relation on scheme $(cname)$.

$$\{t \mid t \in r \text{ borrower} \wedge t[\text{amount}] > 1200\}$$

$$\{t \mid \exists s \in \text{borrower} (t[cname] = s[cname] \wedge s[\text{amount}] > 1200)\}$$

To read this equation as "the set of all tuples t such that there exists a tuple s in the relation *borrower* for which the values of t and s for the *cname* attribute are equal, and the value of s for the *amount* attribute is greater than 1200."

The notation $\exists t \in r Q(t)$ means "there exists a tuple t in relation r such that predicate $Q(t)$ is true".

How did we get the above expression? We needed tuples on scheme *cname* such that there were tuples in *borrower* pertaining to that customer name with *amount* attribute > 1200 .

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

The tuples t get the scheme $cname$ implicitly as that is the only attribute t is mentioned with.

Let's look at a more complex example.

Find all customers having a loan from the Coimbatore branch, and the cities in which they live:

$$\{t \mid \exists s \in \text{borrower}(t[cname]=s[cname] \wedge s[bname]="Coimbatore"$$

$$\wedge t \mid \exists u \in \text{Customer}(u[cname]=s[cname] \wedge t[ccity]=u[ccity])) \}$$

In English, we might read this as "the set of all $(cname,ccity)$ tuples for which $cname$ is a borrower at the SFU branch, and $ccity$ is the city of $cname$ ".

Tuple variable s ensures that the customer is a borrower at the SFU branch.

Tuple variable u is restricted to pertain to the same customer as s , and also ensures that $ccity$ is the city of the customer.

The logical connectives \wedge (AND) and \vee (OR) are allowed, as well as \neg (negation).

We also use the existential quantifier \exists and the universal quantifier \forall .

Some more examples:

1. Find all customers having a loan, an account, or both at the Coimbatore branch:

$$\{t \mid \exists s \in \text{borrower}(t[cname]=s[cname] \wedge s[bname]="Coimbatore"$$

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

$$\wedge \exists u \in \text{depositer}(t[\text{cname}] = u[\text{cname}] \wedge u[\text{bname}] = \text{"Coimbatore"}) \}$$

Note the use of the V connective. As usual, set operations remove all duplicates.

2. Find all customers who have **both** a loan and an account at the Coimbatore branch.

Solution: simply change the V connective in 1 to a \wedge .

2. Find customers who have an account, but **not** a loan at the SFU branch.

$$\{t \mid \exists u \in \text{borrower}(t[\text{cname}] = u[\text{cname}] \wedge u[\text{bname}] = \text{"Coimbatore"})$$

$$\wedge \neg \exists s \in \text{depositer}(t[\text{cname}] = s[\text{cname}] \wedge s[\text{bname}] = \text{"Coimbatore"}) \}$$

4. Find all customers who have an account at **all** branches located in Brooklyn.
(We used **division** in relational algebra.)

For this example we will use implication, denoted by a pointing finger in the text, but by here.

The formula $P \Rightarrow Q$ means P implies Q, or, if P is true, then Q must be true.

$$\{t \mid \forall u \in \text{branch}(u[\text{bcity}] = \text{"Town hall"}) \Rightarrow$$

$$\exists s \in \text{depositer}(t[\text{cname}] = s[\text{cname}] \wedge u[\text{bname}] = s[\text{bname}])) \}$$

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

In English: the set of all *cname* tuples such that for all tuples *u* in the *branch* relation, if the value of on attribute *bcity* is townhall, then the customer has an account at the branch whose name appears in

the *bname* attribute of u.

Formal Definitions

1. A tuple relational calculus expression is of the form

$$\{t \mid P(t)\}$$

where P is a **formula**.

Several tuple variables may appear in a formula.

2. A tuple variable is said to be a **free variable** unless it is quantified by a \forall or \exists .

Then it is said to be a **bound variable**.

3. A formula is built of **atoms**. An atom is one of the following forms:

- $s \in r$, where s is a tuple variable, and r is a relation (\in is not allowed).
- $s[x] \theta u[y]$, where s and u are tuple variables, and x and y are attributes, and θ is a comparison operator ($=, <, >, \leq, \geq, \neq$).
- $S[x] \theta c$, where c is a constant in the domain of attribute x.

4. **Formulae** are built up from atoms using the following rules:

- An atom is a formula.
- If P is a formula, then so are $\neg P$ and (P)
- If P_1 and P_2 are formulae, then so are $P_1 \wedge P_2$, $P_1 \vee P_2$ and $P_1 \neq P_2$.
- If $P(s)$ is a formula containing a free tuple variable s, then

$\forall s (P(s))$ and $\exists s (P(s))$ are formulae.

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: II (Relational Model) BATCH: 2017-2020

are formulae also.

5. Note some equivalences:

- $\sigma_{A \leq B} \sigma_{C \leq D} R = \sigma_{A \leq B \wedge C \leq D} R$
- $\sigma_{A \leq B} \sigma_{C \leq D} R = \sigma_{C \leq D} \sigma_{A \leq B} R$
- $\sigma_{A \leq B} \sigma_{C \leq D} R = \sigma_{A \leq B \vee C \leq D} R$

Safety of Expressions

1. A tuple relational calculus expression may generate an infinite expression, e.g.
2. There are an infinite number of tuples that are not in *borrow*! Most of these tuples contain values that do not appear in the database.

3. Safe Tuple Expressions

We need to restrict the relational calculus a bit.

- The domain of a formula ϕ , denoted $\text{dom}(\phi)$, is the set of all values referenced in ϕ .
- These include values mentioned in ϕ as well as values that appear in a tuple of a relation mentioned in ϕ .

So, the domain of ϕ is the set of all values explicitly appearing in ϕ or that appear in relations mentioned in ϕ .

- $\sigma_{A \leq B} R$ is the set of all values appearing in *borrow*.
- $\sigma_{A \leq B} R$ is the set of all values appearing in *borrow*.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

We may say an expression E is **safe** if all values that appear in the **result** are values from $\text{dom}(A)$.

4. A **safe** expression yields a finite number of tuples as its result. Otherwise, it is called **unsafe**.

Expressive Power of Languages

1. The tuple relational calculus restricted to safe expressions is equivalent in expressive power to the relational algebra.

The Domain Relational Calculus

1. Domain variables take on values from an attribute's domain, rather than values for an entire tuple.

Formal Definitions

1. An expression is of the form

$$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$$

where the x_i represent domain variables, and P is a **formula**.

2. An atom in the domain relational calculus is of the following forms

- $R(x_1, x_2, \dots, x_n)$ where R is a relation on n attributes, and x_1, x_2, \dots, x_n are domain variables or constants.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

- θ , where α and β are domain variables, and θ is a comparison operator.
- $\alpha = c$, where c is a constant.

3. **Formulae** are built up from atoms using the following rules:

- An atom is a formula.
- If α is a formula, then so are $\neg \alpha$ and α .
- If α and β are formulae, then so are $\alpha \wedge \beta$, $\alpha \vee \beta$, and $\alpha \rightarrow \beta$.
- If α is a formula where x is a domain variable, then so are $\exists x \alpha$ and $\forall x \alpha$.

Example Queries

1. Find branch name, loan number, customer name and amount for loans of over \$1200.
2. Find all customers who have a loan for an amount > than \$1200.
3. Find all customers having a loan from the SFU branch, and the city in which they live.
4. Find all customers having a loan, an account or both at the SFU branch.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

5. Find all customers who have an account at **all** branches located in Brooklyn.

— | . . . —

If you find this example difficult to understand, try rewriting this expression using implication, as in the tuple relational calculus example. Here's my attempt:

[— | . . . —]

I've used two letter variable names to get away from the problem of having to remember what stands for.

Safety of Expressions

- As in the tuple relational calculus, it is possible to generate infinite expressions. The solution is similar for domain relational calculus-restrict the form to safe expressions involving values in the **domain** of the formula.

Expressive Power of Languages

- All three of the following are equivalent:
 - The relational algebra.
 - The tuple relational calculus restricted to safe expressions.
 - The domain relational calculus restricted to safe expressions.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: II (Relational Model) **BATCH: 2017-2020**

Possible Questions

PART A (1 MARK)

(Online Examinations)

PART B (2 MARKS)

1. Define- relational algebra.
2. Write short notes on tuple relational calculus.
3. Write short notes on domain relational calculus
4. What is a SELECT operation?
5. What is a PROJECT operation?
6. What are the fundamental operations in relational algebra?
7. Mention the unary and binary operators in relational algebra.
8. List out the type of join operations
9. What is the use of rename operation?
10. What is foreign key?
11. Differentiate between Union and intersection operation.
12. Write short notes on formal definitions of relational algebra.
13. Compare the relational calculus and relational algebra.
14. Define query.
15. Write query for “Find the branch name, loan number and amount for loans of over \$1200” using domain relational calculus.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: II (Relational Model)

BATCH: 2017-2020

PART C (6 Marks)

1. Define Keys. Classify various types of keys with examples.
2. Enlighten on various operators used in relational algebra.
3. Discuss the various type of join operations ? Why are these join required.
4. Describe in detail about tuple and domain relational calculus.
5. Differentiate between Cartesian product and natural join operations used in relational algebra.
6. Consider the following tables:
Employee (Emp_no, Name, Emp_city)
Company (Emp_no, Company_name, Salary)
 - i. Write a SQL query to display Employee name and company name.
 - ii. Write a SQL query to display employee name, employee city ,company name and salary of all the employees whose salary >10000.
 - iii. Write a query to display all the employees working in 'XYZ' company.
7. Explain in detail about tuple relation calculus.
8. Describe in briefly on domain relation calculus.
9. What is Database Schema and explain with examples.
10. Portray on basic structure of relational database.



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-641021
(For candidates admitted from 2017 onwards)

SUBJECT NAME: DATABASE MANAGEMENT SYSTEM

SEMESTER: IV

SUBJECT CODE:17CCU402

UNIT-II

SNO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	The _____ is the responsible picture of relation.	Table	Tuple	Primary key	Attributes	Primary key
2	The _____ is a perceived database used as a collection of normalized relation	Table	Relational database	Predicates	Relation	Relational database
3	A name relation is a _____	Database	Variable	Table	Database	Table
4	The _____ requirement needs a little elaboration	Candidate key	Irreducibility	Foreign key	Nulls	Candidate key
5	The _____ database does not include any invalid foreign key	Referencing	Target relation	Referential integrity	Referenced relation	Referential integrity

6	A relation that contains the corresponding candidate key known as _____	Reference	Target relation	Target tuple	Referential diagram	Target relation
7	The chain of arrow represents _____	Referential tuple	Referential path	Referencing	Target relation	Referential path
8	The component of the piece of the model is called _____	Relational algebra	Project	Closures	Traditional set operations	Relational algebra
9	The foreign key value for which there does not exists a matching value of the candidate keys _____	Unmatched foreign key values	Support for foreign keys	Support for referential integrity	Matched foreign key values	Unmatched foreign key values

10	An _____ are two types of compatible relations.	Union	Interaction	Product	Closure	Union
11	The ____ is a relation A on X,Y,.....Z.	Projection	Product	Closure	Restricti on	Projection
12	The _____ is defined as data to be fetched as a result of retrieval operation.	Retrieval	Securit y rules	Integrity rules	Snapsho ts	Retrieval
13	The defining a data to be inserted is called an _____	Virtual relations	Update	Snapshots	Integrity rules	Integrity rules
14	The defining a data to be made through a view is called as _____	Security rules	Retrieval	Stability requirements	Virtual relations	Security rules
15	The defining a data that is to be the scope of same concurrency is _____	Stability requirements	Update	Integrity roles	Snapshot	Update
16	The attribute value in each tuple is known as ____.	Atomic	Component	Quantity	Null values	Atomic

17	The component of a primary key may be _____.	Quantity	Atomic	Null	Not null	Null
18	The naming structure consists of the relation name plus the name of_____.	Constraints	Extension	Attributes	Tuples	Extension
19	In key constraints are implied by the existence of _____ keys.	Quantity	Attributes	Candidate	Null	Candidate
20	The attributes are ordered in a _____direction.	Left to right	Right to left	Left to left	Right to right	Left to right
21	A each row of a table represents as_____of the relation	N-tuple	M-tuple	D1D2	Nm- tuple	N-tuple
22	The number of tuples in a relation is called_____	Part	Cardinality	Domain	Relation	Cardinality
23	An attribute represents one use of domain within a relation_____	Part	Cardinality	Domain	Relationship	Domain
24	The References are concerned with the _____	Relational calculus	Vertical calculus	Half relational	Cardinality	Relational calculus
25	A relation satisfying the foregoing condition is said to be_____	Normalized	Balanced	Heading	Paragraph	Normalized

26	The combination consists of a _____ attribute	Single	Double	Triple	Fourth	Single
27	The integrity constraints can be subdivided into _____	Constraints	Data	Domain	Function	Constraints
28	A each file contain _____ record.	Two	Three	One	Four	One
29	In each record ,occurrence has a _____	Unique definer	Address	Data	Data dependent	Unique definer
30	The relational algebra is a collection of _____ on relations	Operations	Address	Data	Relation	Operations
31	The union of two relations A and B is__	A Intersect B A	Union B	A Part B	Domain	Union B
32	The intersection of two relations A and B is__	A Intersection B	A Union B	A Part B	A Intersect B	A Intersection B
33	The difference between two relations is A Minus B	A Intersect B	Union B	A Minus B	A Intersect B	A Minus B
34	The extended cartesian product of two relation is _____	A Times B	B Times A	A Minus B	B Minus	A Times B
35	The projection operator yields a _____ subset of a given relation	Horizontal	Vertical	Straight	None	Vertical

36	The division operator divides a _____ relation	Subtraction	Multiplication	Addition	Dividend	Dividend
37	The system that supports relational database but has a language that is powerful than algebra called as _____	Semi relational	Half relational	Full relational	Relational	Semi Relational
38	The host programming language is ____	PL/I	Cobol	C++	Java	PL/I
39	The first embedded SQL document are prefixed by _____ sign	*	\$	#	@	\$
40	The term _____ to mean a select statement for which the retrieval table contains at most one row.	Delete	Create table	Singleton select	Update	Singleton select
41	An _____ increases the status of all london supplier by the amount given by the program variable raise.	Update	Insert	Syr_code	Delete	Update

42	The _____ provide a mechanism for accessing the records in the set one by one.	Cursors	CPU	Monitors	Keyboard	Cursors
43	The executable statements provided to operate on cursors are called _____	Fetch	Select	Insert	Exit	Fetch
44	The _____ of embedded SQL are provided to assist in the process	Hardware	Software	Dynamic	Statement	Software
45	The _____ is a principal dynamic statement	Execute	Select	Open	Fetch	Execute
46	A machine language version of the statement is _____	Prepare	SQL source	SQLOBL	SQLOBJ	SQLOBJ
47	The Projection is denoted by the uppercase greek letter pi _____	II	I	III	IV	II
48	A _____ identifies primary key of another table	Primary Key	Candidate key	Super Key	Foreign Key	Foreign Key
49	The _____ set a quantity and status to zero for all suppliers	Multiple-record updates	Multiple-table updates	Single-record insertion	single table	Multiple-table updates
50	The _____ is a name of all tables to the system.	Retrieval of a table	Retrieval of a column	Multiple e-record in update	Multiple e-record in section	Retrieval of a table

51	The newly stored table is said to be a_____	Discussion	Single record insertion	Snapshot	Fetch	Snapshot
52	A _____ is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.	Primary Key	Candidate key	Super Key	Foreign Key	Super Key
53	The functional dependency is an essential part of understanding the meaning of the _____	Information	Source	Data	Paragraph	Source
54	The normalization leads to a simple means of declaring_____	Functional dependencies	Multiple-record update	Multiple e-record insertion	Single record	Functional dependencies
55	The entities identified by the primary key values are fundamental entities about with data is recorded in the_____	Fetch	File	Database	Record	Database
56	The normalization of given relation is a matter of__	Semantics	Update	Delete	Inversion	Semantics

57	A _____ is uniquely identifies the record in a table.	Primary Key	Candidate key	Super Key	Foreign Key	Primary Key
58	The select, project and rename are called _____ operations	Binary	Unary	Ternary	Single	Unary
59	An executable sql statement can appear _____	Pl/ I statement	Pre compiler	Cobol	SQL	Pl / I statement
60	The data lost on _____ is returned to the program	Delete indication	Save indication	Open indication	Error indication	Error indication

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

UNIT- III
SYLLABUS

Structured Query Language - Basic Structure - Set Operations - Aggregate Functions - Date, Numeric, and Character Functions - Nested Sub queries -Modification Of Databases - Joined Relations-DDL - Embedded SQL.

Structured Query Language

SQL has become *the* standard relational database language. It has several parts:

- Data definition language (DDL) - provides commands to
 - Define relation schemes.
 - Delete relations.
 - Create indices.
 - Modify schemes.
- Interactive data manipulation language (DML) - a query language based on both relational algebra and tuple relational calculus, plus commands to insert, delete and modify tuples.
- Embedded data manipulation language - for use within programming languages like C, PL/1, Cobol, Pascal, etc.
- View Definition - commands for defining views
- Authorization - specifying access rights to relations and views.
- Integrity - a limited form of integrity checking.
- Transaction control - specifying beginning and end of transactions.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

Basic Structure

1. Basic structure of an SQL expression consists of **select**, **from** and **where** clauses.

- **select** clause lists attributes to be copied - corresponds to relational algebra **project**.
- **from** clause corresponds to Cartesian product - lists relations to be used.
- **where** clause corresponds to selection predicate in relational algebra.

2. Typical query has the form

select A_1, A_2, \dots, A_n **from** r_1, r_2, \dots, r_n **where** P

where each A_i represents an attribute, each a r_i relation, and P is a predicate.

3. This is equivalent to the relational algebra expression

$$\pi_{A_1, A_2, \dots, A_n} (\sigma_{P(r_1 \times r_2 \times \dots \times r_m)})$$

- If the where clause is omitted, the predicate P is true.
- The list of attributes can be replaced with a $*$ to select all.
- SQL forms the Cartesian product of the relations named, performs a selection using the predicate, then projects the result onto the attributes named.
- The result of an SQL query is a relation.
- SQL may internally convert into more efficient expressions.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

The select Clause

- An example: Find the names of all branches in the account relation.
select bname from account
 - **distinct vs. all:** elimination or not elimination of duplicates.
 - Find the names of all branches in the account relation.
select distinct bname from account
 - By default, duplicates are not removed. We can state it explicitly using all.
select all bname from account
select * means select all the attributes.
- Arithmetic operations can also be in the selection list.

The where Clause

The predicates can be more complicated, and can involve

- Logical connectives and, or and not.
- Arithmetic expressions on constant or tuple values.
- The between operator for ranges of values.

Example: Find account number of accounts with balances between \$90,000 and \$100,000.

select account# from account where balance between 90000 and 100000

The from Clause

- The from class by itself defines a Cartesian product of the relations in the clause.
- SQL does not have a natural join equivalent. However, natural join can be expressed in terms of a Cartesian product, selection, and projection.

For the relational algebra expression

$$\Pi_{\text{customer-name, loan-number}}(\text{borrower} \bowtie \text{loan})$$

write as

```
select distinct cname, borrower.loan-number from borrower, loan where  
borrower.loan-number = loan.loan-number
```

- More selections with join: Find the names and loan numbers of all customers who have a loan at the SFU branch,
write as

```
select distinct cname, borrower.loan-number from borrower, loan where  
borrower.loan-number = loan.loan-number and bname= "Coimbatore"
```

The Rename Operation

Rename: a mechanism to rename both relations and attributes.

- as-clause can appear in both the select and from clauses:
- old-name as new-name.

Example

```
select distinct cname, borrower.loan-number as loan_id      from borrower, loan  
where borrower.loan-number = loan.loan-number and bname= "Coimbatore"
```

Tuple Variables

- Tuple variables can be used in SQL, and are defined in the from clause:

```
select distinct cname, T.loan-number from borrower as S, loan as T where  
S.loan-number = T.loan-number
```
- The keyword **as** is optional here.

- These variables can then be used throughout the expression. Think of it as being something like the rename operator.

- Finds the names of all branches that have assets greater than at least one branch located in Madurai.

select distinct T.bname from branch S, branch T where S.bcity= "Madurai" and
T.assets > S.assets

String Operations

- The most commonly used operation on strings is pattern matching using the operator like.
- String matching operators % (any substring) and _ (underscore, matching any character).

E.g., ``___%" matches any string with at least 3 characters.

- Patterns are case sensitive, e.g., ``Jim" does not match ``jim".
- Use the keyword escape to define the escape character.
followed by a sequence of characters and then "tely" and then "%\".
- E.g., like "ab%tely \%\" escape "\" matches all the strings beginning with "ab"
- Backslash overrides the special meaning of these symbols.
- We can use not like for string mismatching.
- Example. Find all customers whose street includes the substring ``Main".
select cname from customer where street like "%Main%"
- SQL also permits a variety of strings such as concatenating (using "||"), extracting substrings, finding the length of strings converting between upper case and lower case, and so on.

Ordering the Display of tuples

- SQL allows the user to control the order in which tuples are displayed.
- order by makes tuples appear in sorted order (ascending order by default).
- desc specifies descending order.
- asc specifies ascending order.

Example

Customer ID	Customer name	Contact	Address	City	Postal code	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2		Ana Trujillo	Avda. de la Constitución	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio M. Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.		WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

SELECT * FROM Customers ORDER BY Country DESC;

Customer ID	Customer name	Contact	Address	City	Postal code	Country
1	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
2	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Ana Trujillo Emparedados y	Ana Trujillo	Avda. de la Constitución	México D.F.	05021	Mexico
5	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

Duplicate Tuples

- Formal query languages are based on mathematical relations. Thus no duplicates appear in relations.
- As duplicate removal is expensive, SQL allows duplicates.
- To remove duplicates, we use the **distinct** keyword.
- To ensure that duplicates are not removed, we use the **all** keyword.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

- *Multiset* (bag) versions of relational algebra operators.
 - if there are c_1 copies of tuples t_1 in r_1 , and t_1 satisfies selection σ , then there are c_1 copies of t_1 in $\sigma(r_1)$.
 - for each copy of tuple t_1 in r_1 , there is a copy of tuple $\pi_A(t_1)$ in $\pi_A(r_1)$.
 - if there are c_1 copies of tuple t_1 in r_1 , and c_2 copies of tuple t_2 in r_2 , there is $c_1 \times c_2$ copies of tuple t_1, t_2 in $r_1 \times r_2$.
- An SQL query of the form

select from r_1, r_2, \dots, r_n **where** P is equivalent to the algebra expression

$\pi_{A_1, A_2, \dots, A_n} (\sigma_P (r_1 \times r_2 \times \dots \times r_m))$ **where** P

using the multiset versions of the relational operators σ, π and \times .

Set Operations

1. SQL has the set operations **union**, **intersect** and **except**.
2. Find all customers having an account.

select distinct *cname* **from** *depositor*

union: Find all customers having a loan, an account, or both. branch.

(select *cname* **from** *depositor*) **union** **(select** *cname* **from** *borrower*)

intersect: Find customers having a loan **and** an account.

(select distinct *cname* **from** *depositor*) **intersect** **(select distinct** *cname* **from** *borrower*)

except: Find customers having an account, but **not** a loan.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

(select distinct cname from depositor) except (select cname from borrower)

3. Some additional details:

- **union** eliminates duplicates, being a set operation. If we want to retain duplicates, we may use **union all**, similarly for **intersect** and **except**.
- Not all implementations of SQL have these set operations.
- **except** in SQL-92 is called **minus** in SQL-86.
- It is possible to express these queries using other operations.

SQL Datatypes

Each column in a database table is required to have a name and a data type.

An SQL must decide what type of data that will be stored inside each column when creating a table.

The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.

Three main data types: text, number, and date.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

Text data types:

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. Note: If you put a greater value than 255 it
TINYTEXT	
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted.
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

Number data types:

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

Date data types:

Data type	Description
DATE()	A date. Format: YYYY-MM-DD Note: The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS Note: The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS Note: The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS Note: The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. Note: Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

Aggregate Functions

An aggregate function allows you to perform a calculation on a set of values to return a single scalar value. To use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

The following are the most commonly used SQL aggregate functions:

1. AVG – calculates the average of a set of values.
2. COUNT – counts rows in a specified table or view.
3. MIN – gets the minimum value in a set of values.
4. MAX – gets the maximum value in a set of values.
5. SUM – calculates the sum of values.

These are called **aggregate functions**. They return a single value.

Syntax

aggregate_function (DISTINCT | ALL expression)

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

COUNT function

To get the number of the products in the products table, to use the COUNT function as follows

SELECT COUNT (ProductID) FROM Products;

COUNT(ProductID)
77

AVG function

To calculate the average units in stock of the products, to use the AVG function as follows:

SELECT AVG (Price) FROM Products;

AVG(Price)
28.866363636363637

SUM function

To calculate the sum of units in stock by product category, to use the SUM function with the GROUP BY clause as the following query:

SELECT SUM (Quantity) FROM OrderDetails;

SUM(Quantity)
12743

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

MIN function

To get the minimum units in stock of products in the products table, to use the MIN function as follows:

SELECT MIN (Price) AS SmallestPrice FROM Products;

SmallestPrice
2.5

MAX function

To get the maximum units in stock of products in the products table, to use the MAX function as shown in the following query:

SELECT MAX (Price) AS LargestPrice FROM Products;

LargestPrice
263.5

Date and Time Functions

GETDATE ()

Probably the most essential of the date4 functions,

SELECT GETDATE()

returns a datetime data type containing the current system data and time: 2009-07-07 11:52:26.687.

DATEADD (datepart, number, date)

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

DATEADD lets you add values to a given date and returns the result as a datetime data type.

```
SELECT DATEADD(DAY, 30, GETDATE())
```

adds 30 days to the date from the example above: 2009-08-06 12:01:38.950.

DATEDIFF (datepart, startdate, enddate)

This function returns a single integer data type that represents the difference between two dates. It can return values for years, months, days, hours, minutes, seconds, milliseconds, and more:

```
SELECT DATEDIFF(DAY, '01/01/2009', GETDATE())
```

returns 187 as the difference in days between the example date and the beginning of the year.

DATEPART (datepart, date)

To return an integer that represents a portion of a valid date, DATEPART extracts all parts of the datetime data type including years, months, days, hours, minutes, seconds and milliseconds:

```
SELECT DATEPART(MONTH, GETDATE())
```

returns 7 as the example date's month.

DATENAME (datepart, date)

Like its name suggests, DATENAME returns the name of a given part of the

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

DATE:

```
SELECT DATENAME(MONTH, GETDATE())
```

It can return almost all parts of the date including the name of the quarter, the weekday, or as here, the month: July.

ISDATE (expression)

This function tests if the value supplied is a valid date:

```
SELECT ISDATE ('07/44/09')
```

In this case, it returns a value of 0 (false) indicating the date is invalid; if it returns a value of 1 (true), the date is valid.

DAY(date), MONTH(date), YEAR(date)

These date functions are like DATEPART but a bit easier to work with:

```
SELECT MONTH(0), DAY(0), YEAR(0)
```

They each return an integer representing the supplied date value—in this case, 1,1,1900.

NUMERIC Functions

SQL numeric functions are used primarily for numeric manipulation and/or mathematical calculations.

ABS() function

A mathematical function that returns the absolute (positive) value of the specified numeric expression.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

Function syntax:

ABS(numeric_expression)

ABS Example:

SQL server, MySQL, Oracle, PostgreSQL:

```
select isbn_no, price-35 No_ABS, ABS(price-35) With_ABS from bookstore  
where price-35 <0;
```

The result should be:

isbn_no	no_abs	with_abs
0471777781	-3	3
0596009747	-3	3
1861006314	-6	6

CEIL()/CEILING() FUNCTION

Returns the smallest integer greater than, or equal to, the specified numeric expression.

Function syntax:

CEIL(numeric_expression)

Function syntax:

CEILING(numeric_expression)

CEIL Example1:

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

```
select ISBN_NO, price, price+0.4 NewPrice ,CEIL(price+0.4) NewPrice_CEIL
```

```
from bookstore where price >30;
```

CEILING Example2:

```
select ISBN_NO, price, price+0.4 NewPrice ,CEILING(price+0.4) NewPrice_CEIL
```

```
from bookstore where price >30;
```

The result should be for both examples:

isbn_no	price	newprice	newprice_ceil
0201703092	39	39.4	40
0471777781	32	32.4	33
059600 1747	32	32.4	33
067232 5764	49	49.4	50
076455 7599	42	42.4	43
0764579088	35	35.4	36
186100 1025	38	38.4	39

EXP() FUNCTION

Returns the value of e (the base of natural logarithms) raised to the power of X.

Function syntax:

```
EXP(float_expression )
```

EXP Example:

```
select isbn_no, price , LOG(price ) , EXP(LOG(price )) from bookstore
```

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

where price >34;

EXP Example:

select isbn_no, price , LN(price) , EXP(LN(price)) from bookstore

where price >34;

The result should be for both examples:

isbn_no	price	log(price)	exp(log(price))
0201703092	39	3.66356164612965	39
0672325764	49	3.89182029811063	49
0764557599	42	3.73766961828337	42
0764579088	35	3.55534806148941	35
1861002025	38	3.63758615972639	38

FLOOR() function

Returns the largest integer less than or equal to the specified numeric expression.

Function syntax:

FLOOR(float_expression)

FLOOR Example:

select isbn_no, price+0.6 No_FLOOR, FLOOR(price+0.6) With_FLOOR from bookstore

where price >34;

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

The result should be:

isbn_no	no_floor	with_floor
0201703092	39.6	39
0672325764	49.6	49
0764557599	42.6	42
0764579088	35.6	35
1861002025	38.6	38

LN()/LOG() FUNCTION

Returns the natural logarithm of the specified float expression. The inverse of this function is EXP().

LNfunction syntax:

LN(float_expression)

LOG() function syntax:

LOG(float_expression [, base])

LOG() function syntax:

LOG([base,] float_expression)

LN Example1:

```
select isbn_no, price , LN(price ) , EXP(LN(price )) from bookstore
```

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

where price >34;

The result should be:

isbn_no	price	ln(price)	exp(ln(price))
0201703092	39	3.66356164612965	39
0672325764	49	3.89182029811063	49
0764557599	42	3.73766961828337	42
0764579088	35	3.55534806148941	35
1861002025	38	3.63758615972639	38

LN Example2:

select isbn_no, price , LOG(price) , EXP(LOG(price)) from bookstore

where price >34;

The result should be:

isbn_no	price	log(price)	exp(log(price))
0201703092	39	3.66356164612965	39
0672325764	49	3.89182029811063	49
0764557599	42	3.73766961828337	42
0764579088	35	3.55534806148941	35
1861002025	38	3.63758615972639	38

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

POWER() function

Returns the value of the specified expression to the specified power.

Function syntax:

POWER(float_expression ,power)

POWER Example:

```
SELECT isbn_no, item_qty, price, POWER(price,2) NewPrice,  
item_qty*POWER(price,2) NewCost FROM order_books;
```

The result should be:

isbn_no	item_qty	price	newprice	newcost
0201703092	2	39	1521	3042
0764557599	1	42	1764	1764
1861002025	5	38	1444	7220
1861006314	2	29	841	1682
0672325764	1	49	2401	2401
0764557599	2	42	1764	3528
0764579038	4	35	1225	4900
0201703092	10	39	1521	15210
0471777781	2	32	1024	2048
1861002025	5	38	1444	7220

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

1861006314	1	29	841	841
------------	---	----	-----	-----

ROUND() function

Returns a numeric value, rounded to the specified length or precision.

Function syntax:

ROUND(numeric_expression , length)

Length is the precision to which numeric_expression is to be rounded. Length can be negative to cause Length digits left of the decimal point of the numeric_expression to become zero.

ROUND Example:

```
SELECT isbn_no, price, price*1.077,ROUND( price*1.077,1) NewPrice
```

```
FROM bookstore where price >0;
```

The result should be for both examples:

isbn_no	price	price*1.077	newprice
02017C3092	39	42.003	42.0
0471777781	32	34.464	34.5
05960C9747	32	34.464	34.5
0672325764	49	52.773	52.8
0764557599	42	45.234	45.2
0764579088	35	37.695	37.7

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

1861002025	38	40.926	40.9
1861006314	29	31.233	31.2

SIGN() function

Returns the positive (+1), zero (0), or negative (-1) sign of a numeric expression.

Function syntax:

SIGN(numeric_expression)

SIGN Example:

```
select isbn_no, price, price-35, SIGN(price-35) from bookstore
```

```
where price >0;
```

The result should be:

isbn_no	price	price-35	sign(price-35)
0201700092	39	4	1
0471777781	32	-3	-1
0596009747	32	-3	-1
0672325764	49	14	1

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

0764557599	42	7	1
0764579088	35	0	0
1861002025	38	3	1
1861006314	29	-6	-1

SQRT() function

Returns the square root of a nonnegative numeric expression.

Function syntax:

SQRT(numeric_expression)

SQRT Example:

SELECT isbn_no, item_qty, price,

ROUND(SQRT(price*price+20*20),1) NewPrice

FROM order_books;

The result should be for both examples:

isbn_no	item_qty	price	newprice
0201703092	2	39	43.8
0764557599	1	42	46.5
1861002025	5	38	42.9
1861006314	2	29	35.2

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

0672325764	1	49	52.9
0764557599	2	42	46.5
0764579088	4	35	40.3
0201703092	10	39	43.8
0471777781	2	32	37.7
1861002025	5	38	42.9
1861006314	1	29	35.2

Character Functions

SQL provides a rich set of character functions that allow you to get information about strings and modify the contents of those strings in multiple ways. Character functions are of the following two types:

1. Case-Manipulative Functions (LOWER, UPPER and INITCAP)
2. Character-Manipulative Functions (CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE)

Case-Manipulative Functions

1. **LOWER** : This function converts alpha character values to lowercase. LOWER will actually return a fixed-length string if the incoming string is fixed-length. LOWER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign (\$) or modulus (%).

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

2. Syntax:

LOWER(SQL course)

Input1: SELECT LOWER('GEEKSFORGEEKS') FROM DUAL;

Output1: geeksforgeeks

Input2: SELECT LOWER('DATABASE@456') FROM DUAL;

Output2: database@456

3. **UPPER :** This function converts alpha character values to uppercase. Also UPPER function too, will actually return a fixed-length string if the incoming string is fixed-length. UPPER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign (\$) or modulus (%).

Syntax:

UPPER(SQL course)

Input1: SELECT UPPER('geeksforgeeks') FROM DUAL;

Output1: GEEKSFORGEEKS

Input2: SELECT UPPER('dbms\$508%7') FROM DUAL;

Output2: DBMS\$508%7

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

4. **INITCAP** : This function converts alpha character values to uppercase for the first letter of each word and all others in lowercase. The words in the string must be separated by either # or _ or space.

Syntax:

INITCAP(SQL course)

Input1: SELECT INITCAP('geeksforgeeks is a computer science portal for geeks')
FROM DUAL;

Output1: Geeksforgeeks Is A Computer Science Portal For Geeks

Input2: SELECT INITCAP('PRACTICE_CODING_FOR_EFFICIENCY') FROM DUAL;

Output2: Practice_Coding_For_Efficiency

Character-Manipulative Functions

1. **CONCAT** : This function always appends (concatenates) string2 to the end of string1. If either of the string is NULL, CONCAT function returns the non-NULL argument. If both strings are NULL, CONCAT returns NULL.

Syntax:

CONCAT('String1', 'String2')

Input1: SELECT CONCAT('computer' , 'science') FROM DUAL;

Output1: computerscience

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

Input2: SELECT CONCAT(NULL ,'Android') FROM DUAL;

Output2: Android

Input3: SELECT CONCAT(NULL ,NULL) FROM DUAL;

Output3: -

2. **LENGTH :** This function returns the length of the input string. If the input string is NULL, then LENGTH function returns NULL and not Zero. Also, if the input string contains extra spaces at the start, or in between or at the end of the string, then the LENGTH function includes the extra spaces too and returns the complete length of the string.

Syntax:

LENGTH(Column | Expression)

Input1: SELECT LENGTH('Learning Is Fun') FROM DUAL;

Output1: 15

Input2: SELECT LENGTH(' Write an Interview Experience ') FROM DUAL;

Output2: 34

Input3: SELECT LENGTH('') FROM DUAL; or SELECT LENGTH(NULL) FROM DUAL;

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

Output3: -

- 3. SUBSTR :** This function returns a portion of a string from a given start point to an end point. If a substring length is not given, then SUBSTR returns all the characters till the end of string (from the starting position specified).

Syntax:

SUBSTR('String',start-index,length_of_extracted_string)

Input1: SELECT SUBSTR('Database Management System', 9) FROM DUAL;

Output1: Management System

Input2: SELECT SUBSTR('Database Management System', 9, 7) FROM DUAL;

Output2: Manage

- 4. INSTR :** This function returns numeric position of a character or a string in a given string. Optionally, you can provide a position m to start searching, and the occurrence n of string. Also, if the starting position is not given, then it starts search from index 1, by default. If after searching in the string, no match is found then, INSTR function returns 0.

Syntax: INSTR(Column | Expression, 'String', [,m], [n])

Input: SELECT INSTR('Google apps are great applications','app',1,2) FROM DUAL;

Output: 23

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

5. **LPAD and RPAD** : These functions return the strings padded to the left or right (as per the use) ; hence the “L” in “LPAD” and the “R” in “RPAD” ; to a specified length, and with a specified pad string. If the pad string is not specified, then the given string is padded on the left or right (as per the use) with spaces.

Syntax:

LPAD(Column | Expression, n, 'String')

Syntax: **RPAD(Column | Expression, n, 'String')**

LPAD Input1: SELECT LPAD('100',5,*) FROM DUAL;

LPAD Output1: **100

LPAD Input2: SELECT LPAD('hello', 21, 'geek') FROM DUAL;

LPAD Output2: geekgeekgeekgeekhello

RPAD Input1: SELECT RPAD('5000',7,*) FROM DUAL;

RPAD Output1: 5000***

RPAD Input1: SELECT RPAD('earn', 19, 'money') FROM DUAL;

RPAD Output1: earnmoneymoneymoney

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

6. **TRIM** : This function trims the string input from the start or end (or both). If no string or char is specified to be trimmed from the string and there exists some extra space at start or end of the string, then those extra spaces are trimmed off.

Syntax:

TRIM(Leading|Trailing|Both, trim_character FROM trim_source)

Input1: SELECT TRIM('G' FROM 'GEEKS') FROM DUAL;

Output1: EEKS

Input2: SELECT TRIM(' geeksforgeeks ') FROM DUAL;

Output2:geeksforgeeks

7. **REPLACE** : This function searches for a character string and, if found, replaces it with a given replacement string at all the occurrences of the string. REPLACE is useful for searching patterns of characters and then changing all instances of that pattern in a single function call. If a replacement string is not given, then REPLACE function removes all the occurrences of that character string in the input string. If neither a match string nor a replacement string is specified, then REPLACE returns NULL.

Syntax:

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

REPLACE(Text, search_string, replacement_string)

Input1: SELECT REPLACE('DATA MANAGEMENT', 'DATA','DATABASE') FROM DUAL;

Output1: DATABASE MANAGEMENT

Input2: SELECT REPLACE('abcdeabcccabddddeabcc', 'abc') FROM DUAL;

Output2: deccabddddeec

Nested subqueries

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow –

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

Subqueries with the SELECT Statement

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows –

```
SELECT column_name [, column_name ] FROM table1 [, table2 ]
WHERE column_name OPERATOR (SELECT column_name [, column_name ]
FROM table1 [, table2 ] [WHERE])
```

Example

Consider the CUSTOMERS table having the following records –

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, let us check the following subquery with a SELECT statement.

```
SQL> SELECT * FROM CUSTOMERS WHERE ID IN (SELECT ID FROM
CUSTOMERS WHERE SALARY > 4500) ;
```

This would produce the following result.

ID	NAME	AGE	DDRESS	SALARY	
----	------	-----	--------	--------	--

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

4	Chaitali	25	Mumbai	6500.00	
5	Hardik	27	Bhopal	8500.00	
7	Muffy	24	Indore	10000.00	

Subqueries with the INSERT Statement

Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. The selected data in the subquery can be modified with any of the character, date or number functions.

The basic syntax is as follows.

```
INSERT INTO table_name [ (column1 [, column2 ]) ] SELECT [ * | column1 [, column2 ] ] FROM table1 [, table2 ]
```

[WHERE VALUE OPERATOR]

Example

Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table. Now to copy the complete CUSTOMERS table into the CUSTOMERS_BKP table, you can use the following syntax.

```
SQL> INSERT INTO CUSTOMERS_BKP SELECT * FROM CUSTOMERS WHERE ID IN (SELECT ID FROM CUSTOMERS);
```

Subqueries with the UPDATE Statement

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

The basic syntax is as follows.

```
UPDATE table
```

```
SET column_name = new_value
```

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

[WHERE OPERATOR [VALUE]

(SELECT COLUMN_NAME

FROM TABLE_NAME)

[WHERE)]

Example

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table. The following example updates SALARY by 0.25 times in the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27.

SQL> UPDATE CUSTOMERS

SET SALARY = SALARY * 0.25

WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP

WHERE AGE >= 27);

This would impact two rows and finally CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	hmedabad	125.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	2125.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	0000.00

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

Subqueries with the DELETE Statement

The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above.

The basic syntax is as follows.

DELETE FROM TABLE_NAME

[WHERE OPERATOR [VALUE]

(SELECT COLUMN_NAME

FROM TABLE_NAME)

[WHERE)]

Example

Assuming, we have a CUSTOMERS_BKP table available which is a backup of the CUSTOMERS table. The following example deletes the records from the CUSTOMERS table for all the customers whose AGE is greater than or equal to 27.

SQL> DELETE FROM CUSTOMERS

WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP

WHERE AGE >= 27);

This would impact two rows and finally the CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)

BATCH: 2017-2020

Modification of Database

The SQL Modification Statements make changes to database data in tables and columns. There are 3 modification statements:

- INSERT Statement -- add rows to tables
- UPDATE Statement -- modify columns in table rows
- DELETE Statement -- remove rows from tables

INSERT Statement

The INSERT Statement adds one or more rows to a table. It has two formats:

INSERT INTO table-1 [(column-list)] VALUES (value-list)

and,

INSERT INTO table-1 [(column-list)] (query-specification)

The first form inserts a single row into table-1 and explicitly specifies the column values for the row. The second form uses the result of query-specification to insert one or more rows into table-1. The result rows from the query are the rows added to the insert table. Note: the query cannot reference table-1.

Both forms have an optional column-list specification. Only the columns listed will be assigned values. Unlisted columns are set to null, so unlisted columns must allow nulls. The values from the VALUES Clause (first form) or the columns from the query-specification rows (second form) are assigned to the corresponding column in column-list in order.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

If the optional column-list is missing, the default column list is substituted. The default column list contains all columns in table-1 in the order they were declared in CREATE TABLE, or CREATE VIEW.

VALUES Clause

The VALUES Clause in the INSERT Statement provides a set of values to place in the columns of a new row. It has the following general format:

VALUES (value-1 [, value-2] ...)

value-1 and value-2 are Literal Values or Scalar Expressions involving literals. They can also specify NULL.

The values list in the VALUES clause must match the explicit or implicit column list for INSERT in degree (number of items). They must also match the data type of corresponding column or be convertible to that data type.

INSERT Examples

INSERT INTO p (pno, color) VALUES ('P4', 'Brown')

Before

pno	descr	color
P1	Widget	Blue
P2	Widget	Red

=>

After

pno	descr	color
P1	Widget	Blue
P2	Widget	Red

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

P3	Dongle	Green

P3	Dongle	Green
P4	NULL	Brown

INSERT INTO sp SELECT s.sno, p.pno, 500 FROM s, p WHERE p.color='Green' AND s.city='London'

Before

After

sno	pno	qty
S1	P1	NULL
S2	P1	200
S3	P1	1000
S3	P2	200

=>

sno	pno	qty
S1	P1	NULL
S2	P1	200
S3	P1	1000
S3	P2	200
S2	P3	500

UPDATE Statement

The UPDATE statement modifies columns in selected table rows. It has the following general format:

UPDATE table-1 SET set-list [WHERE predicate]

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

The optional WHERE Clause has the same format as in the SELECT Statement. See WHERE Clause. The WHERE clause chooses which table rows to update. If it is missing, all rows in table-1 are updated.

The set-list contains assignments of new values for selected columns. See SET Clause.

The SET Clause expressions and WHERE Clause predicate can contain subqueries, but the subqueries cannot reference table-1. This prevents situations where results are dependent on the order of processing.

SET Clause

The SET Clause in the UPDATE Statement updates (assigns new value to) columns in the selected table rows. It has the following general format:

SET column-1 = value-1 [, column-2 = value-2] ...

column-1 and column-2 are columns in the Update table. value-1 and value-2 are expressions that can reference columns from the update table. They also can be the keyword -- NULL, to set the column to null.

Since the assignment expressions can reference columns from the current row, the expressions are evaluated first. After the values of all Set expressions have been computed, they are then assigned to the referenced columns. This avoids results dependent on the order of processing.

UPDATE Examples

UPDATE sp SET qty = qty + 20

Before

After

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

sno	pno	qty
S1	P1	NULL
S2	P1	200
S3	P1	1000
S3	P2	200

=>

sno	pno	qty
S1	P1	NULL
S2	P1	220
S3	P1	1020
S3	P2	220

UPDATEs

SET name = 'Tony', city

Before

sno	name	city
S1	Pierre	Paris
S2	John	London
S3	Mario	Rome

sno	name	city
S1	Pierre	Paris
S2	John	London
S3	Tony	Milan

DELETE Statement

The DELETE Statement removes selected rows from a table. It has the following general format:

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

DELETE FROM table-1 [WHERE predicate]

The optional WHERE Clause has the same format as in the SELECT Statement.

See WHERE Clause. The WHERE clause chooses which table rows to delete. If it is missing, all rows in table-1 are removed.

The WHERE Clause predicate can contain subqueries, but the subqueries cannot reference table-1. This prevents situations where results are dependent on the order of processing.

DELETE Examples

DELETE FROM sp WHERE pno = 'P1'

Before

sno	pno	qty
S1	P1	NULL
S2	P1	200
S3	P1	1000
S3	P2	200

=>

After

sno	pno	qty
S3	P2	200

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

DELETE FROM p WHERE pno NOT IN (SELECT pno FROM sp)

Before

pno	descr	color
P1	Widget	Blue
P2	Widget	Red
P3	Dongle	Green

=>

After

pno	descr	color
P1	Widget	Blue
P2	Widget	Red

Joined Relations

The basic Cartesian-product mechanism for joining tuples of relations provided by earlier versions of SQL, SQL-92 also provides various mechanisms for joining relations, including condition joins and natural joins as well as various forms of outer joins.

These additional operations are typically used a subquery expressions in the from clause.

Examples

1. Two given relations: *loan* and *borrower*.

Bname	Loan #	Amount
Downtown	L-170	3000
Redwood	L-230	7000
Perryridge	L-260	1700

Ename	Loan#
Jones	L-170
Smith	L-230

The *loan* and *borrower* relations.

inner join:

loan **inner join** *borrower* **on** *loan.loan-number = borrower.loan-number*

Notice that the loan-number will appear twice in the inner joined relation.

Bname	Loan #	Amount	Ename	Loan#
Downtown	L-170	3000	Jones	L-170
Redwood	L-230	7000	Smith	L-230

Result of *loan* inner join *borrower*.

left outer join:

loan **left outer join** *borrower* **on** *loan.loan# = borrower.loan#*

Bname	Loan #	Amount	Ename	Loan#
Downtown	L-170	3000	Jones	L-170
Redwood	L-230	7000	Smith	L-230
Perryridge	L-260	1700	null	null

Result of *loan* left outer join *borrower*

natural inner join:

loan **natural inner join** *borrower*

Bname	Loan #	Amount	Ename
Downtown	L-170	3000	Jones
Redwood	L-230	7000	Smith

Result of *loan* natural inner join *borrower*.

Join types and conditions

1. Each variant of the join operations in SQL-92 consists of a *join type* and a *join condition*.
2. Join types: **inner join**, **left outer join**, **right outer join**, **full outer join**.

The keyword **inner** and **outer** are optional since the rest of the join type enables us to deduce whether the join is an inner join or an outer join.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

SQL-92 also provides two other join types:

1. **cross join**: an inner join without a join condition.
2. **union join**: a full outer join on the ``false" condition, i.e., where the inner join is empty.
3. Join conditions: **natural**, **on** predicate, **using** $A_1, A_2, A_3, \dots, A_n$.

The use of join condition is mandatory for outer joins, but is optional for inner joins (if it is omitted, a Cartesian product results).

Ex. A natural full outer join:

loan **natural full outer join** *borrower* **using** (loan-number)

Bname	Loan #	Amount	Ename
Downtown	L-170	3000	Jones
Redwood	L-230	7000	Smith
Perryridge	L-260	1700	null
null	L-155	null	Hayes

Result of *loan* natural full outer join *borrower* **using** (loan#).

Example. Find all customers who have either an account or a loan (but not both) at the bank.

select cname from (natural full outer join borrower) where account# is null or loan# is null

Data-Definition Language

The SQL DDL (Data Definition Language) allows specification of not only a set of relations, but also the following information for each relation:

- The schema for each relation.
- The domain of values associated with each attribute.
- Integrity constraints.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

- The set of indices for each relation.
- Security and authorization information.
- Physical storage structure on disk.

Domain Types in SQL

1. The SQL-92 standard supports a variety of built-in domain types:
 - **char**(*n*) (or **character**(*n*)): fixed-length character string, with user-specified length.
 - **varchar**(*n*) (or **character varying**): variable-length character string, with user-specified maximum length.
 - **int** or **integer**: an integer (length is machine-dependent).
 - **smallint**: a small integer (length is machine-dependent).
 - **numeric**(*p*, *d*): a fixed-point number with user-specified precision, consists of *p* digits (plus a sign) and *d* of *p* digits are to the right of the decimal point. E.g., **numeric**(3, 1) allows 44.5 to be stored exactly but not 444.5.
 - **real** or **double precision**: floating-point or double-precision floating-point numbers, with machine-dependent precision.
 - **float**(*n*): floating-point, with user-specified precision of at least *n* digits.
 - **date**: a calendar date, containing four digit year, month, and day of the month.
 - **time**: the time of the day in hours, minutes, and seconds.
2. SQL-92 allows arithmetic and comparison operations on various numeric domains, including, **interval** and *cast (type coercion)* such as transforming between *smallint* and *int*. It considers strings with different length are compatible types as well.
3. SQL-92 allows **create domain** statement, e.g.,

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

create domain *person-name* **char**(20)

Schema definition in SQL

1. An SQL relation is defined by:

create table *r* ($A_1D_1, A_2D_2, A_3D_3, \dots, A_nD_n$, (*integrity-constraint*₁) ,..., *integrity-constraint*_k)

where *r* is the relation name, A_i is the name of an attribute, and D_i is the domain of that attribute. The allowed integrity-constraints include

primary key ($A_{j1}, A_{j2}, \dots, A_{jm}$) and **check**(*P*)

Example.

create table *branch* (*bname* **char**(15) **not null** *bcity* **char**(30) *assets* **integer**
primary key (*bname*) **check** (*assets* >= 0))

2. The values of primary key must be *not null* and *unique*. SQL-92 consider **not null** in primary key specification is redundant but SQL-89 requires to define it explicitly.

3. Check creates type checking functionality which could be quite useful. E.g.,

create table *student* (*name* **char**(15) **not null** *student-id* **char**(10) **not null** *degree-level* **char**(15) **not null** **check** (*degree-level* **in** ("Bachelors", "Masters", "Doctorate")))

4.

Some checking (such as *foreign-key* constraints) could be costly, e.g.,

check (*bname* **in** (**select** *bname* **from** *branch*))

5. A newly loaded table is empty. The **insert** command can be used to load it, or use special bulk loader utilities.

6. To remove a relation from the database, we can use the **drop table** command:

drop table *r*

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

This is not the same as **delete** *r* which retains the relation, but deletes all tuples in it.

7. The **alter table** command can be used to add or drop attributes to an existing relation *r*:

alter table *r* **add** *A D* where *A* is the attribute and *D* is the domain to be added.

alter table *r* **drop** *A* where *A* is the attribute to be dropped.

Embedded SQL

1. SQL provides a powerful declarative query language. However, access to a database from a general-purpose programming language is required because,
 - SQL is not as powerful as a general-purpose programming language. There are queries that cannot be expressed in SQL, but can be programmed in C, Fortran, Pascal, Cobol, etc.
 - Nondeclarative actions -- such as printing a report, interacting with a user, or sending the result to a GUI -- cannot be done from within SQL.
2. The SQL standard defines embedding of SQL as *embedded SQL* and the language in which SQL queries are embedded is referred as *host language*.
3. The result of the query is made available to the program one tuple (record) at a time.
4. To identify embedded SQL requests to the preprocessor, we use EXEC SQL statement:
5. EXEC SQL embedded SQL statement END-EXEC

Note: A semi-colon is used instead of END-EXEC when SQL is embedded in C or Pascal.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

6. Embedded SQL statements: **declare cursor**, **open**, and **fetch** statements.

EXEC SQL

declare *c* **cursor for**

select *cname, ccity*

from *deposit, customer*

where *deposit.cname = customer.cname and deposit.balance > :amount*

END-EXEC

where *amount* is a host-language variable.

EXEC SQL **open** *c* END-EXEC

This statement causes the DB system to execute the query and to save the results within a temporary relation.

A series of **fetch** statement are executed to make tuples of the results available to the program.

EXEC SQL **fetch** *c into* *:cn, :cc* END-EXEC

The program can then manipulate the variable *cn* and *cc* using the features of the host programming language.

A single **fetch** request returns only one tuple. We need to use a **while** loop (or equivalent) to process each tuple of the result until no further tuples (when a variable in the SQLCA is set).

We need to use **close** statement to tell the DB system to delete the temporary relation that held the result of the query.

EXEC SQL **close** *c* END-EXEC

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

7. Embedded SQL can execute any valid **update**, **insert**, or **delete** statements.
8. *Dynamic* SQL component allows programs to construct and submit SQL queries at run time.
9. SQL-92 also contains a *module* language, which allows procedures to be defined in SQL.

CLASS: II B.COM CA
COURSE CODE: 17CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: III (SQL)
BATCH: 2017-2020

POSSIBLE QUESTIONS

PART A (1 Mark)

(Online Examinations)

PART B (2 Marks)

1. List the set operations of SQL.
2. List the string operations supported by SQL.
3. What are aggregate functions? And list the aggregate functions supported by SQL?
4. List the table modification commands in SQL.
5. List the SQL domain Types.
6. What is view in SQL? How is it defined?
7. What is the use of group by clause?
8. Give the general form of SQL query.
9. What are the categories of SQL command?
10. What are the parts of SQL language?
11. What are various Data types in SQL ?
12. What is the use of sub queries?
13. Write the syntax of date and time functions.
14. What is meant by joined relations.
15. How are the nulls represented in database system?

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 17CCU402

UNIT: III (SQL)

BATCH: 2017-2020

PART C (6 Marks)

1. Explain the aggregate functions used in relational algebra.
2. Discuss briefly about Embedded SQL.
3. Portray on set operation with examples.
4. Describe on Nested Sub queries in DBMS.
5. Write in detail about Data- Definition Language.
6. Illustrate on joined relations with examples.
7. List out the operations in modification of database and explain with examples.
8. Discuss about the basic structure of SQL.
9. Depict on several parts of SQL Language.
10. What are the Character functions in SQL? Explain it.



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-641021
(For candidates admitted from 2017 onwards)

SUBJECT NAME: DATABASE MANAGEMENT SYSTEM
SUBJECT CODE:17CCU402

SEMESTER: IV

UNIT-III

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	A value of zero indicates that the sql statement executed_____	Failure	Successfully	Balance	Null	Successfully
2	A value of _____ indicates that no data was found in the database that satisfied the request.	100	200	11	90	100
3	A _____ might be used to save an intermediate result	Create table	Pre-compiler	Save indication	Pre-compiler	Create table
4	An embedded sql provides a bridge by means of a new type of object called a_____	Keyboard	CPU	Monitor	Cursor	Cursor
5	An operation does not involve cursor is _____	Exit	Single to select	Open	Select	Single to select

6	The _____ retrieve status and city for the supplier.	Single to n select	Update	Insert	Delete	Single to n select
7	The _____ is used to set for the negative value.	Sys_Code	Sql	Select	Update	Sys_Code
8	A cursor is defined by means of _____ statement, which takes in general form.	Let	Cobol	C++	Java	Let
9	The LET statement is used for _____	Executable	Balanced	Negative	Declarative	Declarative
10	The sets of records that are associated with a cursor are always considered to have a _____ Clause.	Position	Line	Paragraph	Ordering	Ordering
11	The _____ statement is normally executed with in a program loop	Fetch	Open	Insert	Delete	Fetch
12	The first sql statement executed _____ in a program	\$Begin transaction	\$End transaction	\$Continue Transaction	\$Fetch	\$Begin transaction

13	The last sql statement executed _____ in a program	\$Begin transaction	\$End transaction	\$Continue transaction	\$Fetch	\$end transaction
14	The _____ is used to receive feedback information	PL/I Structure	SQL Statement	Open	Exit	PL/I structure
15	The _____ provides certain features to facilitate the writing of online application program.	Sql	Basic	Cobol	Java	Sql
16	A _____ of embedded sql are provided to assists the process	Let statement	Dynamic statement	SQL statement	Fetch statement	Dynamic statement
17	In the sql _____ actually uses the keyword	Form	Select	Open	Exit	Form
18	The _____ is probably much more convenient to construct dynamically.	Dynamic statement	Sql statement	Discussion	Transaction	Sql statement

19	The _____ statements can be replaced by another by issuing	Select	Open	Delete	Prepare	Prepare
20	The term _____ mean to exclude the cursor statements	Open	Close	Regular	Delete	Regular
21	The dynamically generated statement does not contain _____	Reference	Parameters	Open	Fetch	Reference
22	The _____ statement requires special treatment	Save	Delete	Close	Select	Select
23	The DO stands for _____	Order	Paragraph	Descending order	Ascending order	Descending order
24	The AO stands for _____	Order	Paragraph	Descending order	Ascending order	Ascending order
25	The _____ symbol is used to insert operator	I	J	O	K	I
26	The _____ symbol is used to delete operator	A	B	C	D	D
27	The operations are not involving in cursors _____	Select, Insert, Update, Delete	Open, fetch, ,close	Open and fetch	delete	Select, Insert, Update, Delete

28	The database design is _____ concerned	Intention	Extension	Relation	Legal extension	Intention
29	The _____ is mean intentional part of the relational rather than the extensional part .	Combination	Result	Relation	Extension	Relation
30	A _____ is a mechanism for accessing the records in the set one by one	Cursor	Management	Normal Form	Relation	Cursor
31	A _____ query is defined by means of a Statement	Open	Let	Close	Fetch	Let
32	The _____ query is used for executable Statement	Insert	Satisfactory	Balanced	Not balanced	Insert
33	The _____ statement is used to get part numbers to form all parts supplied.	Simple Retrieval	Qualified Retrieval	Retrieval with ordering	Retrieval using a link	Retrieval using a link

34	The _____ query is used to defining relation schemas, deleting relations and modifying relation schemas.	DDL	DML	TCL	DCL	DDL
35	The _____ query is used to insert tuples, delete tuples and modify tuples in the database	DDL	DML	TCL	DCL	DML
36	The DDL statement for specifies _____	SQL	DML	DDL	Integrity Constraints	Integrity Constraints
37	The _____ statement control is used for specifying the begin and end of the transaction.	Embedded	DML	DDL	Transaction	Transaction
38	The _____ view consists of a collection of the physical database.	Conceptual	Record	Tree	Field	Conceptual

39	The term _____ is context since the user does not store a database exactly.	Physical	Secondary	Hierarchical	Relational	Physical
40	An each physical database is defined by a _____	Database record	Database description	Database view	Database field	Database description
41	A _____ is a fixed length character string with user-specified length n.	Character	Char(n)	Int	Float	Char(n)
42	A _____ is a fixed point number with user-specified precision.	Number	Number (p)	Numeric (p,d)	Number(d)	Numeric(p, d)
43	A _____ is a floating point number with the precision of atleast n digits.	Int	Float	Float(n)	Char	Float(n)

44	The___ is a variable length character string with user-specified maximum length n.	Char(n)	Variable(n)	Varchar(n)	Int(n)	Varchar(n)
45	The remove a relation from an sql database___command is used.	Delete table	Alter table	Drop table	Create table	Drop table
46	The_____clause is particularly used in defining the notion of tuple variables.	From	As	Like	Order by	As
47	The___clause tuples of a query to appear in sorted order	As	Order by	Like	From	Order by
48	The_____operation cannot be eliminating duplicates automatically	Union	Intersect	Order by	Except	Order by
49	The___command is used to delete tuples from a relation	Alter	Update	Delete	Drop	Delete

50	The _____ clause corresponds to the projection operation of the relational algebra.	Select	From	Where	Drop	Select
51	The _____ clause corresponds to the cartesian product operation of the relational algebra.	Select	From	Where	Drop	From
52	The _____ clause corresponds to the selection predicate of the relational algebra.	Select	From	Where	Drop	Where
53	The _____ command for specifying access relations and views	View definition	Authorization	Dynamic sql	Embedded sql	Authorization
54	The _____ command is used to add attributes to an existing relation.	Create table	Alter table	Drop table	Update table	Alter table

55	The sql expresses patterns by using the _____ comparison operator	Escape	Like	Upper	Not like	Like
56	The _____ function that take a collection of values as input and return a single value.	Input	Output	Aggregate	Sum	Aggregate
57	The attribute given in the ____ clause are used to form groups.	Where	Group by	Select	Update	Group by
58	The _____ function is used to find the number of values	Count	Sum	Avg	Max	Count
59	The _____ function is used to find the sum of the values	COUNT	SUM	AVG	MAX	SUM
60	The lowercase greek letter _____ is used to denote selection operation	Sigma	Pi	Union	Intersect	Sigma

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

UNIT- IV

SYLLABUS

Relational Database Design - Pitfalls - Normalisation Using Functional Dependencies
- First Normal Form-Second Normal Form-Third Normal Form-Fourth Normal Form
And BCNF.

Pitfalls in Relational Database Design

Relational database design requires that we find a “good” collection of relation schemas. A bad design may lead to

- Repetition of Information.
- Inability to represent certain information.

Design Goals:

- Avoid redundant data
- Ensure that relationships among attributes are represented
- Facilitate the checking of updates for violation of database integrity constraints.

Example

Consider the relation schema:

Lending-schema = (branch-name, branch-city, assets, customer-name, loan-number, amount)

branch-	branch-	assets	customer-	loannumber	amount
---------	---------	--------	-----------	------------	--------

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

name	city		name		
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

Problems:

Redundancy:

- Data for branch-name, branch-city, assets are repeated for each loan that a branch makes

Wastes space:

- Complicates updating, introducing possibility of inconsistency of assets value

Null values

- Cannot store information about a branch if no loans exist
- Can use null values, but they are difficult to handle.

What is functional dependency?

- Functional Dependency is a relationship that exists between multiple attributes of a relation.
- This concept is given by E. F. Codd.
- Functional dependency represents a formalism on the infrastructure of relation.
- It is a type of constraint existing between various attributes of a relation.
- It is used to define various normal forms.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

- These dependencies are restrictions imposed on the data in database.
- If P is a relation with A and B attributes, a functional dependency between these two attributes is represented as $\{A \rightarrow B\}$. It specifies that,

A	It is a determinant set.
B	It is a dependent attribute.
$\{A \rightarrow B\}$	A functionally determines B. B is a functionally dependent on A.

- Each value of A is associated precisely with one B value. A functional dependency is trivial if B is a subset of A.
- 'A' Functionality determines 'B' $\{A \rightarrow B\}$ (Left hand side attributes determine the values of Right hand side attributes).

For example: <Employee> Table

EmpId	EmpName

- In the above <Employee> table, EmpName (employee name) is functionally dependent on EmpId (employee id) because the EmpId is unique for individual names.
- The EmpId identifies the employee specifically, but EmpName cannot distinguish the EmpId because more than one employee could have the same name.
- The functional dependency between attributes eliminates the repetition of information.

- It is related to a candidate key, which uniquely identifies a tuple and determines the value of all other attributes in the relation.

Advantages of Functional Dependency

- Functional Dependency avoids data redundancy where same data should not be repeated at multiple locations in same database.
- It maintains the quality of data in database.
- It allows clearly defined meanings and constraints of databases.
- It helps in identifying bad designs.
- It expresses the facts about the database design.

Introduction to Axioms Rules

- Armstrong's Axioms is a set of rules.
- It provides a simple technique for reasoning about functional dependencies.
- It was developed by William W. Armstrong in 1974.
- It is used to infer all the functional dependencies on a relational database.

Various Axioms Rules

A. Primary Rules

Rule 1	Reflexivity	If A is a set of attributes and B is a subset of A, then A holds B. $\{A \rightarrow B\}$
Rule 2	Augmentation	If A hold B and C is a set of attributes, then AC holds BC. $\{AC \rightarrow BC\}$ It means that attribute in dependencies does not change the basic dependencies.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

Rule 3	Transitivity	<p>If A holds B and B holds C, then A holds C.</p> <p>If $\{A \rightarrow B\}$ and $\{B \rightarrow C\}$, then $\{A \rightarrow C\}$</p> <p>A holds B $\{A \rightarrow B\}$ means that A functionally determines B.</p>
--------	--------------	---

B. Secondary Rules

Rule 1	Union	<p>If A holds B and A holds C, then A holds BC.</p> <p>If $\{A \rightarrow B\}$ and $\{A \rightarrow C\}$, then $\{A \rightarrow BC\}$</p>
Rule 2	Decomposition	<p>If A holds BC and A holds B, then A holds C.</p> <p>If $\{A \rightarrow BC\}$ and $\{A \rightarrow B\}$, then $\{A \rightarrow C\}$</p>
Rule 3	Pseudo Transitivity	<p>If A holds B and BC holds D, then AC holds D.</p> <p>If $\{A \rightarrow B\}$ and $\{BC \rightarrow D\}$, then $\{AC \rightarrow D\}$</p>

Sometimes Functional Dependency Sets are not able to reduce if the set has following properties,

1. The Right-hand side set of functional dependency holds only one attribute.
2. The Left-hand side set of functional dependency cannot be reduced, it changes the entire content of the set.
3. Reducing any functional dependency may change the content of the set.

A set of functional dependencies with the above three properties are also called as Canonical or Minimal.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

Trivial Functional Dependency

Trivial	If A holds B $\{A \rightarrow B\}$, where A is a subset of B, then it is called a Trivial Functional Dependency. Trivial always holds Functional Dependency.
Non-Trivial	If A holds B $\{A \rightarrow B\}$, where B is not a subset A, then it is called as a Non-Trivial Functional Dependency.
Completely Non-Trivial	If A holds B $\{A \rightarrow B\}$, where $A \cap Y = \Phi$, it is called as a Completely Non-Trivial Functional Dependency.

Example:

Consider relation $E = (P, Q, R, S, T, U)$ having set of Functional Dependencies (FD).

$P \rightarrow Q$ $P \rightarrow R$
 $QR \rightarrow S$ $Q \rightarrow T$
 $QR \rightarrow U$ $PR \rightarrow U$

Calculate some members of Axioms are as follows,

1. $P \rightarrow T$
2. $PR \rightarrow S$
3. $QR \rightarrow SU$
4. $PR \rightarrow SU$

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

Solution:

1. $P \rightarrow T$

In the above FD set, $P \rightarrow Q$ and $Q \rightarrow T$

So, Using Transitive Rule: If $\{A \rightarrow B\}$ and $\{B \rightarrow C\}$, then $\{A \rightarrow C\}$

\therefore If $P \rightarrow Q$ and $Q \rightarrow T$, then $P \rightarrow T$.

$P \rightarrow T$

2. $PR \rightarrow S$

In the above FD set, $P \rightarrow Q$

As, $QR \rightarrow S$

So, Using Pseudo Transitivity Rule: If $\{A \rightarrow B\}$ and $\{BC \rightarrow D\}$, then $\{AC \rightarrow D\}$

\therefore If $P \rightarrow Q$ and $QR \rightarrow S$, then $PR \rightarrow S$.

$PR \rightarrow S$

3. $QR \rightarrow SU$

In above FD set, $QR \rightarrow S$ and $QR \rightarrow U$

So, Using Union Rule: If $\{A \rightarrow B\}$ and $\{A \rightarrow C\}$, then $\{A \rightarrow BC\}$

\therefore If $QR \rightarrow S$ and $QR \rightarrow U$, then $QR \rightarrow SU$.

$QR \rightarrow SU$

4. $PR \rightarrow SU$

So, Using Pseudo Transitivity Rule: If $\{A \rightarrow B\}$ and $\{BC \rightarrow D\}$, then $\{AC \rightarrow D\}$

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

\therefore If $PR \rightarrow S$ and $PR \rightarrow U$, then $PR \rightarrow SU$.

$PR \rightarrow SU$

Decomposition

- Decomposition is the process of breaking down in parts or elements.
- It replaces a relation with a collection of smaller relations.
- It breaks the table into multiple tables in a database.
- It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.
- If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

Properties of Decomposition

1. Lossless Decomposition
2. Dependency Preservation
3. Lack of Data Redundancy

1. Lossless Decomposition

Decomposition must be lossless. It means that the information should not get lost from the relation that is decomposed.

It gives a guarantee that the join will result in the same relation as it was decomposed.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

Example:

Let's take 'E' is the Relational Schema, With instance 'e'; is decomposed into: E1, E2, E3, En; With instance: e1, e2, e3, en, If $e1 \bowtie e2 \bowtie e3 \dots \bowtie en$, then it is called as 'Lossless Join Decomposition'.

In the above example, it means that, if natural joins of all the decomposition give the original relation, then it is said to be lossless join decomposition.

Example: <Employee_Department> Table

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Bangalore	25000	D005	Human Resource

Decompose the above relation into two relations to check whether a decomposition is lossless or lossy.

Now, we have decomposed the relation that is Employee and Department.

Relation 1 : <Employee> Table

Eid	Ename	Age	City	Salary
E001	ABC	29	Pune	20000
E002	PQR	30	Pune	30000

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Bangalore	25000

Employee Schema contains (Eid, Ename, Age, City, Salary).

Relation 2 : <Department> Table

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

Department Schema contains (Deptid, Eid, DeptName).

So, the above decomposition is a Lossless Join Decomposition, because the two relations contains one common field that is 'Eid' and therefore join is possible.

Now apply natural join on the decomposed relations.

Employee ⋈ Department

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

E005	STU	32	Bangalore	25000	D005	Human Resource
------	-----	----	-----------	-------	------	-------------------

Hence, the decomposition is Lossless Join Decomposition.

If the <Employee> table contains (Eid, Ename, Age, City, Salary) and <Department> table contains (Deptid and DeptName), then it is not possible to join the two tables or relations, because there is no common column between them. And it becomes Lossy Join Decomposition.

2. Dependency Preservation

- Dependency is an important constraint on the database.
- Every dependency must be satisfied by at least one decomposed table.
- If $\{A \rightarrow B\}$ holds, then two sets are functional dependent. And, it becomes more useful for checking the dependency easily if both sets in a same relation.
- This decomposition property can only be done by maintaining the functional dependency.
- In this property, it allows to check the updates without computing the natural join of the database structure.

3. Lack of Data Redundancy

- Lack of Data Redundancy is also known as a Repetition of Information.
- The proper decomposition should not suffer from any data redundancy.
- The careless decomposition may cause a problem with the data.
- The lack of data redundancy property may be achieved by Normalization process.

Introduction to Normalization

- Normalization is a process of organizing the data in the database.
- It is a systematic approach of decomposing tables to eliminate data redundancy.
- It was developed by E. F. Codd.
- Normalization is a multi-step process that puts the data into a tabular form by removing the duplicate data from the relation tables.
- It is a step by step decomposition of complex records into simple records.
- It is also called as Canonical Synthesis.
- It is the technique of building database structures to store data.

Definition of Normalization

“Normalization is a process of designing a consistent database by minimizing redundancy and ensuring data integrity through decomposition which is lossless.”

Features of Normalization

- Normalization avoids the data redundancy.
- It is a formal process of developing data structures.
- It promotes the data integrity.
- It ensures data dependencies make sense that means data is logically stored.
- It eliminates the undesirable characteristics like Insertion, Updation and Deletion Anomalies.

Types of Normalization

1. First Normal Form
2. Second Normal Form
3. Third Normal Form

CLASS: II B.COM CA **COURSE NAME: DATABASE MANAGEMENT SYSTEM**
COURSE CODE: 16CCU402 **UNIT: IV (Relational Database Design)** **BATCH: 2016-2019**

4. Fourth Normal Form
5. BCNF (Boyce – Codd Normal Form)

1. First Normal Form (1NF)

- First Normal Form (1NF) is a simple form of Normalization.
- It simplifies each attribute in a relation.
- In 1NF, there should not be any repeating group of data.
- Each set of column must have a unique value.
- It contains atomic values because the table cannot hold multiple values.

Example:

Employee

Table

ECode	Employee_Name	Department_Name
1	ABC	Sales, Production
2	PQR	Human Resource
3	XYZ	Quality Assurance, Marketing

Employee

Table

using

1NF

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

ECode	Employee_Name	Department_Name
1	ABC	Sales
1	ABC	Production
2	PQR	Human Resource
3	XYZ	Quality Assurance
3	XYZ	Marketing

2. Second Normal Form (2NF)

- In 2NF, the table is required in 1NF.
- The main rule of 2NF is, *'No non-prime attribute is dependent on the proper subset of any candidate key of the table.'*
- An attribute which is not part of candidate key is known as non-prime attribute.

Example : Employee Table using 1NF

ECode	Employee_Name	Employee_Age
1	ABC	38

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

1	ABC	38
2	PQR	38
3	XYZ	40
3	XYZ	40

Candidate

Key: ECode,

Employee_Name

Non prime attribute: Employee_Age

- The above table is in 1NF. Each attribute has atomic values. However, it is not in 2NF because non prime attribute Employee_Age is dependent on ECode alone, which is a proper subset of candidate key. This violates the rule for 2NF as the rule says 'No non-prime attribute is dependent on the proper subset of any candidate key of the table'.

2NF (Second Normal Form) : Employee1 Table

ECode	Employee_Age
1	38
2	38
3	40

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

Employee2 Table

ECode	Employee_Name
1	ABC
1	ABC
2	PQR
3	XYZ
3	XYZ

- Now, the above tables comply with the Second Normal Form (2NF).

3. Third Normal Form (3NF)

- Third Normal Form (3NF) is used to minimize the transitive redundancy.
- In 3NF, the table is required in 2NF.
- While using the 2NF table, there should not be any transitive partial dependency.
- 3NF reduces the duplication of data and also achieves the data integrity.

Example : <Employee> Table

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

EId	Ename	DOB	City	State	Zip
001	ABC	10/05/1990	Pune	Maharashtra	411038
002	XYZ	11/05/1988	Mumbai	Maharashtra	400007

- In the above <Employee> table, EId is a primary key but City, State depends upon Zip code.
- The dependency between Zip and other fields is called Transitive Dependency.
- Therefore we apply 3NF. So, we need to move the city and state to the new <Employee_Table2> table, with Zip as a Primary key.

<Employee_Table1> Table

EId	Ename	DOB	Zip
001	ABC	10/05/1990	411038
002	XYZ	11/05/1988	400007

<Employee_Table2> Table

City	State	Zip

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

Pune	Maharashtra	411038
Mumbai	Maharashtra	400007

- The advantage of removing transitive dependency is, it reduces the amount of data dependencies and achieves the data integrity.
- In the above example, using with the 3NF, there is no redundancy of data while inserting the new records.
- The City, State and Zip code will be stored in the separate table. And therefore the updation becomes more easier because of no data redundancy.

4. BCNF (Boyce – Code Normal Form)

- BCNF which stands for Boyce – Code Normal Form is developed by Raymond F. Boyce and E. F. Codd in 1974.
- BCNF is a higher version of 3NF.
- It deals with the certain type of anomaly which is not handled by 3NF.
- A table complies with BCNF if it is in 3NF and any attribute is fully functionally dependent that is $A \rightarrow B$. (Attribute 'A' is determinant).
- If every determinant is a candidate key, then it is said to be BCNF.
- Candidate key has the ability to become a primary key. It is a column in a table.

Example : <EmployeeMain> Table

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

Empid	Ename	DeptName	DepType
E001	ABC	Production	D001
E002	XYZ	Sales	D002

The functional dependencies are:

Empid → EmpName

DeptName → DeptType

Candidate Key:

Empid

DeptName

- The above table is not in BCNF as neither Empid nor DeptName alone are keys.
- We can break the table in three tables to make it comply with BCNF.

<Employee> Table

Empid	EmpName
E001	ABC
E002	XYZ

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

<Department> Table

DeptName	DeptType
Production	D001
Sales	D002

<Emp_Dept> Table

Empid	DeptName
E001	Production
E002	Sales

Now, the functional dependencies are:

Empid → EmpName

DeptName → DeptType

Candidate Key:

<Employee> Table : Empid

<Department> Table : DeptType

<Emp_Dept> Table : Empid, DeptType

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

- So, now both the functional dependencies left side part is a key, so it is in the BCNF.

5. Fourth Normal Form (4NF)

- Fourth Normal Form (4NF) does not have non-trivial multivalued dependencies other than a candidate key.
- 4NF builds on the first three normal forms (1NF, 2NF and 3NF) and the BCNF.
- It does not contain more than one multivalued dependency.
- This normal form is rarely used outside of academic circles.

For example : A table contains a list of three things that is 'Student', 'Teacher', 'Book'. Teacher is in charge of Student and recommended book for each student. These three elements (Student, Teacher and Book) are independent of one another. Changing the student's recommended book, for instance, has no effect on the student itself. This is an example of multivalued dependency, where an item depends on more than one value. In this example, the student depends on both teacher and book.

- Therefore, 4NF states that a table should not have more than one dependencies.
- While performing 5NF, the table must be in 4NF.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

POSSIBLE QUESTIONS

PART A (1 Mark)

(Online Examinations)

PART B (2 Marks)

1. What is Functional Dependency?
2. What do you mean by redundancy ?How this can be avoided ?
3. What do you understand by dependency preservation?
4. What is transitive dependency?
5. What is lossy decomposition?
6. What is the necessity of normalization?
7. What is normalization?
8. What is normal form?
9. What are the different normal forms?
10. What is the dependency preservation property for decomposition?
11. Write the importance's of decomposition?
12. Define Boyce codd normal form.
13. List the pitfalls in Relational Database Design.
14. Define lossy-less decomposition.

CLASS: II B.COM CA

COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 16CCU402

UNIT: IV (Relational Database Design)

BATCH: 2016-2019

PART C (6 Marks)

1. Explain 2NF and 3NF in detail.
2. Portray on normalization using functional dependencies.
3. Explain 1NF, 2NF and BCNF with suitable example.
4. What are the pitfalls in relational database design? With a suitable example, explain the role of functional dependency in the process of normalization.
5. What is third normalization? Explain third normal forms with example.
6. Give a relation R with four attributes $R = \{A, B, C, D\}$ and the following FD, identify the candidate keys for R and the highest normal forms.
7. Why normalization needed? What are its disadvantages?
8. Discuss the various normal form in normalization with suitable examples.
9. Define term anomalies. Explain BCNF in detail.
10. Explain the concepts of join dependency and lossless decomposition.



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-641021
(For candidates admitted from 2017 onwards)

SUBJECT NAME: DATABASE MANAGEMENT SYSTEM

SEMESTER: IV

SUBJECT CODE:17CCU402

UNIT-IV

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	A functional dependency is a special form of ----- ---	Integrity constraints	Functionally determines	Functionally dependent	Prepare	Integrity constraints
2	A BCNF is conceptually _____ than 3NF	Difficult	Simpler	Balanced	Not balanced	Simpler
3	The normalization of given relation is a matter of-----	Semantics	Update	Delete	Inversion	Semantics
4	The first normal is used to_____	Avoids repetition	Remove repetition	Delete repetition	Accept repetition	Remove repetition
5	The normalization leads to a simple means of declaring__	Functional dependencies	Multiple-record updates	Multiple-records insertion	Multiple record -deletion	Functional dependencies
6	The FD stands for____	Functional dependency	Facilitate dependency	Functional data	Facilitate independence	Functional dependency
7	The_____is a group of dependencies	Facilitate	Non-functional	Trivial functional	Normalization	Trivial functional

8	The Functional dependencies are the types of constraints that are based on _____	Keys	Key revisited	Superset key	Primary key	Keys
9	The BCNF stands for _____	Boyce-Codd Normal Forms	Boce-Codd Normal Forms	Boyce- C Normal Forms	B- Codd Normal Forms	Boyce-Codd Normal Forms

10	In the _____ normal form, a composite attribute is converted to individual attributes.	First	Second	Third	Fourth	First
11	A table on the many sides of a one to many and many to many relationships must be in_____	Second Normal Form (2NF)	Third Normal Form (3NF)	Third Normal Form (3NF)	Composite key	Composite key
12	A Table in second normal form is used to_____	Eliminate all hidden dependencies	Eliminate the possibility of an insertion anomalies	Composite key	Non-key fields depend on the whole primary key	Eliminate all hidden dependencies
13	Which one of the following statements about normal forms is false?	BCNF is stricter than 3 NF	Lossless, dependency - preserving decomposition into 3 NF is always possible	Lossless, dependency – preserving decomposition into BCNF is always possible	Any relation with two attributes is BCNF	Lossless, dependency – preserving decomposition into BCNF is always possible
14	The _____ is a bottom-up approach to design a database	Functional dependency	Database modeling	Normalization	Decomposition	Normalization
15	Which forms are based on minimal data aggregates and repetitive groups?	1NF	2NF	3NF	4NF	3NF

16	Which forms are based on the concept of functional Dependency?	1NF	2NF	3NF	4NF	3NF
17	In relation schema ,more than one key values can be classified as _____	Foreign key	Super key	Candidate key	Primary key	Candidate key
18	The form of dependency in the set of attributes that are neither a subset of any keys nor the candidate key is _____	Transitive dependency	Full functional dependency	Partial dependency	Primary functional dependency	Transitive dependency
19	The process of analyzing relation schemas to achieve minimal redundancy is _____	Normalization of data	Denomination of data	Isolation of data	De-normalization of data	Normalization of data
20	A relational database, the functional dependency between two attributes A and B is denoted by _____ relation	$A \rightarrow B$	$B \leftarrow A$	$AB \rightarrow R$	$R \leftarrow AB$	$A \rightarrow B$

21	The procedure of storing higher normal form relations which are in lower normal form as a base relation is classified as _____	Isolation of data	De-normalization of data	Normalization of data	Denomination of data	De-normalization of data
22	A reflexive rule the dependencies are generated _____	Trivial	Non-trivial	Inferential	Functional	Trivial
23	The constraint of functional dependencies are consider as _____	Legal extensions	Semantic extensions	State extension	Relation extensions	Legal extensions
24	In normalization of relation, the property which is critical and must be achieved is _____	Non-additive join property	Additive join property	Independency reservation property	Dependency preservation property	Non-additive join property
25	In the functional dependency between two sets of attributes A and B then the set of attribute A is indicates as _____	Top- right side	Down-left side	Left-hand side	Right-hand side	Left-hand side

26	The functional dependency in which removal of some attributes does not affect dependency is called as _____	Fully functional dependency	Partial dependency	Prime functional dependency	Transitive dependency	Partial dependency
27	The constraint between two different attributes sets is classified as _____	Modification anomaly	Functional dependency	Insertion anomaly	Deletion anomaly	Functional dependency
28	The normalization form which is based on the transitive dependency is classified as _____	First normal form	Second normal form	Fourth normal form	Third normal form	Third normal form
29	The functional dependency in which removal from some attributes must affect dependency is called as-----	Fully functional dependency	Partial dependency	Prime functional dependency	Transitive dependency	Fully functional dependency
30	The _____ normal form only includes single atomic values	Third normal form	First normal form	Second normal form	Fourth normal form	First normal form

31	The fully-functional dependency is classified as _____	Fourth normal form	Third normal form	First normal form	Second normal form	Second normal form
32	An unnormalized form relation contain as _____ values.	Atomic	Non- Atomic	Classified	Unique values	Classified
33	A relation scheme is _____ form the values are atomic.	Unnormalized	First normal	Boyce - Codd Normal forms	Second normal	First normal
34	A second normal form does not permit _____ dependency between a non-prime attribute and the relation key.	Partial	Multi	Functional	Valued	Partial
35	The Fifth Normal form is concerned with _____	Functional dependency	Multivalued dependency	Join dependency	Domain-key	Join dependency
36	The 2NF is used to__	No functional dependencies exist.	No multivalued dependencies exist	No partial functional dependencies exist	No partial multivalued dependencies exist.	No partial functional dependencies exist

37	The normalization was first proposed by_____	Code	Codd	Boyce Codd	Boyce	Codd
38	The functional dependency exist $A \rightarrow B$, $DB \rightarrow C$ implies_____	$DA \rightarrow C$	$A \rightarrow C$	$B \rightarrow A$	$DB \rightarrow A$	$DA \rightarrow C$
39	Which normal form is considered adequate for normal relational database design?	2NF	5NF	4NF	3NF	3NF
40	Which of the following is not a consequence of non-normalized database?	Update Anomaly	Insertion Anomaly	Redundancy	Lost update problem	Lost update problem
41	The dependency preservation is not guaranteed in_____	BCNF	3NF	PJNF	DKNF	BCNF
42	The normalization theory is built around the concept of _____	Normal forms	Functional dependency	Boyce	INF	Functional dependency
43	The fundamental notion of functional dependencies is_____	FD	FC	FP	FS	FD
44	A functional dependencies is a special form of_____	Integrity constraint	Constraint	Referential integrity	Normal forms	Integrity constraint

45	A_____ form is based on functional dependency	First normal	Second Normal	Seven Normal	Functional dependency	First Normal
46	The MVDs stands for _____	Multivalued dependency	Multi dependency	Maximum valued dependency	Multiple valued dependency	Multivalued dependency
47	The PD stands for _____	Partial dependency	Prime dependency	Partial dependent	Proper dependency	Partial dependency
48	The TD stand for _____	Trivial dependency	Transitive dependency	Travel dependency	Transaction dependency	Transitive dependency
49	The JD stands for _____	Join dependencies	Joint dependencies	Join dependent	Join decomposing	Join dependencies
50	The techniques used for functional dependencies is _____	Axioms	Anomalies	Redundancy	Normal forms	Axioms
51	In_____normal form an attribute column of a table cannot hold multiple values.	First	Second	Third	BCNF	First
52	A_____ is the process of removing redundant data from the table in order to improve storage efficiency, data integrity and scalability	Partial Functional Dependency	Dependency	Normalization	Fully dependency	Normalization

53	The ____ is to lack of data for insertion such that null values in keys should be avoided	Insert Anomaly	Update Anomaly	External Anomaly	Deletion Anomaly	Insert Anomaly
54	A ____ key is composed multiple columns.	Foreign	Candidate	Composite	Super	Composite
55	The ____ non-key column is dependent on some, but not all the columns in a composite key.	Partial functional dependency	Dependency	Fully functional dependency	Partial functional dependency and dependency	Partial functional dependency
56	A ____ is a type of functional dependency in which the value in a non-key column is determined by the value in another non-key column.	Fully functional dependency	Transitive dependency	Dependency	Partial functional dependency	Transitive Dependency
57	The ____ value of one column is dependent on another column.	Fully functional dependency	Functional dependency	Partial functional dependency	Dependency	Functional dependency
58	A table is in BCNF it is in 3NF and every determinant is a ____ key	Dependent	Normal	Candidate	Dependent and normal	Candidate

59	The _____ normal form has a relation that possesses data about an individual entity	2 NF	3NF	4NF	5 NF	4NF
60	The primary key is selected from the _____	Foreign Key	Super key	Candidate Key	Composite key	Candidate Key

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

UNIT- V

SYLLABUS

Oracle - Introduction - SQL (DDL,DML, DCL Commands) - Integrity Constraints - PL/SQL - PL/SQL Block - procedure, function - Cursor management - Triggers - Exception Handling.

What is Oracle

- ORACLE is a fourth generation relational database management system.
- In general, a database management system (DBMS) must be able to reliably manage a large amount of data in a multi-user environment so that many users can concurrently access the same data.
- A DBMS must also be secure from unauthorized access and provide efficient solutions for failure recovery.
- The ORACLE Server provides efficient and effective solutions for the major database features.
- ORACLE consists of many tools that allow to create an application with ease and flexibility.

Different editions of Oracle database

Enterprise Edition: It is the most robust and secure edition. It offers all features, including superior performance and security.

Standard Edition: It provides the base functionality for users that do not require Enterprise Edition's robust package.

Express Edition (XE): It is the lightweight, free and limited Windows and Linux edition.

Oracle Lite: It is designed for mobile devices.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

The Oracle Corporation

Oracle Corporation is the largest software company in the field of database business. Its relational database was the first to support SQL which has since become the industry standard.

Oracle database is one of the most trusted and widely used relational database engines. The biggest rival of Oracle database is Microsoft's SQL Server.

History of Oracle

Oracle was originally developed by Lawrence Ellison (Larry Ellison) and his two friends and former co-worker in 1977. Oracle DB runs on the most major platforms like Windows, UNIX, Linux and Mac OS.

Oracle Database deployment

Oracle Database may be licensed and deployed on-premises on a choice of platforms including Oracle Engineered Systems, and on-Cloud with a choice of services running on general purpose hardware or Exadata. The various editions and Cloud services provide different levels of database functionality for different use cases (e.g. dev/test, departmental and non-critical apps, mission-critical workloads) with different levels of performance, availability, etc. service levels.

- Oracle Database Cloud Services: for on-Cloud and Cloud at Customer deployments
- Oracle Database editions: mainly for on-prem deployments
- Engineered systems such as Oracle Exadata specifically built for Oracle Database deployment (on-prem, on-Cloud, Cloud at Customer)

SQL

SQL language is divided into four types of primary language statements: DML, DDL, DCL and TCL. Using these statements, we can define the structure of a database by

CLASS: II B.COM CA
COURSE CODE: 16CCU402

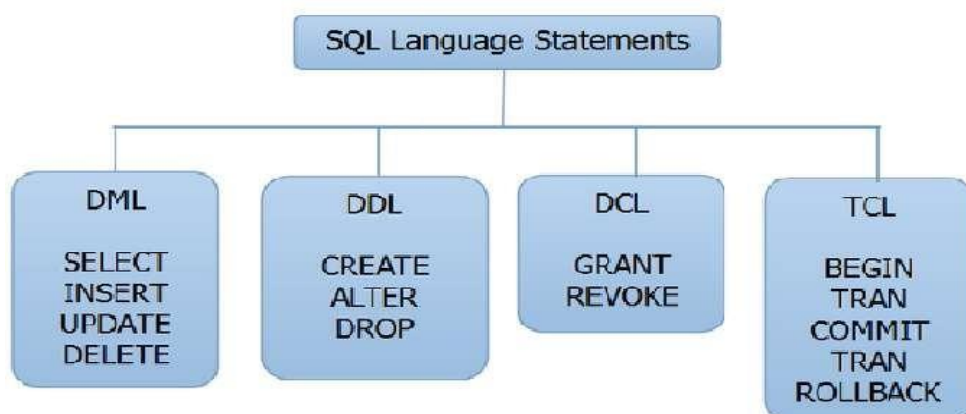
COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

creating and altering database objects, and we can manipulate data in a table through updates or deletions. We also can control which user can read/write data or manage transactions to create a single unit of work.

The four main categories of SQL statements are as follows:

1. DML (Data Manipulation Language)
2. DDL (Data Definition Language)
3. DCL (Data Control Language)
4. TCL (Transaction Control Language)



DML (Data Manipulation Language)

DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

DML statements include the following:

SELECT – select records from a table

INSERT – insert new records

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

UPDATE – update/Modify existing records

DELETE – delete existing records

DDL (Data Definition Language)

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

CREATE – create a new Table, database, schema

ALTER – alter existing table, column description

DROP – delete existing objects from database

DCL (Data Control Language)

DCL statements control the level of access that users have on database objects.

GRANT – allows users to read/write on certain database objects

REVOKE – keeps users from read/write permission on database objects

TCL (Transaction Control Language)

TCL statements allow to control and manage transactions to maintain the integrity of data within SQL statements.

BEGIN Transaction – opens a transaction

COMMIT Transaction – commits a transaction

ROLLBACK Transaction – ROLLBACK a transaction in case of any error

PL/SQL -Introduction

PL/SQL is a procedural language extension to SQL, making it extremely simple to write procedural code that includes SQL as if it were a single language. In

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

comparison, most other programming languages require mapping data types, preparing statements and processing result sets, all of which require knowledge of specific APIs.

The data types in PL/SQL are a super-set of those in the database, so we rarely need to perform data type conversions when using PL/SQL. Ask every average Java or .NET programmer how they find handling date values coming from a database. They can only wish for the simplicity of PL/SQL.

INTEGRITY CONSTRAINTS OVER RELATION

- Database integrity refers to the validity and consistency of stored data. Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate. Constraints may apply to each attribute or they may apply to relationships between tables.
- Integrity constraints ensure that changes (update deletion, insertion) made to the database by authorized users do not result in a loss of data consistency. Thus, integrity constraints guard against accidental damage to the database.

EXAMPLE- A blood group must be 'A' or 'B' or 'AB' or 'O' only (can not any other values else).

TYPES OF INTEGRITY CONSTRAINTS

Various types of integrity constraints are-

1. Domain Integrity
2. Entity Integrity Constraint
3. Referential Integrity Constraint
4. Key Constraints

1. Domain Integrity-

Domain integrity means the definition of a valid set of values for an attribute. To define data type, length or size, is null value allowed, is the value unique or not for an attribute, the default value, the range (values in between) and/or specific values for the attribute.

2. Entity Integrity Constraint-

This rule states that in any database relation value of attribute of a primary key can't be null.

EXAMPLE- Consider a relation "STUDENT" Where "Stu_id" is a primary key and it must not contain any null value whereas other attributes may contain null value e.g "Branch" in the following relation contains one null value.

Stu_id	Name	Branch
11255234	Aman	CSE
11255369	Kapil	EcE
11255324	Ajay	ME
11255237	Raman	CSE
11255678	Aastha	ECE

3. Referential Integrity Constraint-

It states that if a foreign key exists in a relation then either the foreign key value must match a primary key value of some tuple in its home relation or the foreign key value must be null.

The rules are:

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

1. Can't delete a record from a primary table if matching records exist in a related table.
2. Can't change a primary key value in the primary table if that record has related records.
3. Can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.
4. However, to enter a Null value in the foreign key, specifying that the records are unrelated.

EXAMPLE-

Consider 2 relations "stu" and "stu_1" Where "Stu_id" is the primary key in the "stu" relation and foreign key in the "stu_1" relation.

Relation "stu"

Stu_id	Name	Branch
11255234	Aman	CSE
11255369	Kapil	EcE
11255324	Ajay	ME
11255237	Raman	CSE
11255678	Aastha	ECE

Relation "stu_1"

Stu_id	Course	Duration
---------------	---------------	-----------------

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Stu_id	Course	Duration
11255234	B TECH	4 years
11255369	B TECH	4 years
11255324	B TECH	4 years
11255237	B TECH	4 years
11255678	B TECH	4 years

Examples

Rule 1. Can't delete any of the rows in the "stu" relation that are visible since all the "stu" are in use in the "stu_1" relation.

Rule 2. Can't change any of the "Stu_id" in the "stu" relation since all the "Stu_id" are in use in the "stu_1" relation. **Rule 3.** The values that can enter in the "Stu_id" field in the "stu_1" relation must be in the "Stu_id" field in the "stu" relation.

Rule 4 To enter a null value in the "stu_1" relation if the records are unrelated.

4.Key Constraints-

A Key Constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple. The types of key constraints-

- Primary key constraints
- Unique key constraints
- Foreign Key constraints
- NOT NULL constraints
- Check constraints

1. Primary key constraints

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Primary key is the term used to identify one or more columns in a table that make a row of data unique. Although the primary key typically consists of one column in a table, more than one column can comprise the primary key.

For example, either the employee's Social Security number or an assigned employee identification number is the logical primary key for an employee table. The objective is for every record to have a unique primary key or value for the employee's identification number. Because there is probably no need to have more than one record for each employee in an employee table, the employee identification number makes a logical primary key. The primary key is assigned at table creation.

The following example identifies the EMP_ID column as the PRIMARY KEY for the EMPLOYEES table:

```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID          CHAR(9)          NOT NULL PRIMARY KEY,
EMP_NAME        VARCHAR (40)      NOT NULL,
EMP_ST_ADDR     VARCHAR (20)      NOT NULL,
EMP_CITY        VARCHAR (15)      NOT NULL,
EMP_ST          CHAR(2)           NOT NULL,
EMP_ZIP         INTEGER(5)        NOT NULL,
EMP_PHONE       INTEGER(10)       NULL,
EMP_PAGER       INTEGER(10)       NULL);
```

2. Unique Constraints

A unique column constraint in a table is similar to a primary key in that the value in that column for every row of data in the table must have a unique value. Although a

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

primary key constraint is placed on one column, to place a unique constraint on another column even though it is not actually for use as the primary key.

CREATE TABLE EMPLOYEE_TBL

(EMP_ID	CHAR(9)	NOT NULL	PRIMARY KEY,
EMP_NAME	VARCHAR (40)	NOT NULL,	
EMP_ST_ADDR	VARCHAR (20)	NOT NULL,	
EMP_CITY	VARCHAR (15)	NOT NULL,	
EMP_ST	CHAR(2)	NOT NULL,	
EMP_ZIP	INTEGER(5)	NOT NULL,	
EMP_PHONE	INTEGER(10)	NULL	UNIQUE,
EMP_PAGER	INTEGER(10)	NULL)	

3. Foreign Key Constraints

A foreign key is a column in a child table that references a primary key in the parent table. A foreign key constraint is the main mechanism used to enforce referential integrity between tables in a relational database. A column defined as a foreign key is used to reference a column defined as a primary key in another table.

CREATE TABLE EMPLOYEE_PAY_TBL

(EMP_ID	CHAR(9)	NOT NULL,
POSITION	VARCHAR2(15)	NOT NULL,
DATE_HIRE	DATE	NULL,
PAY_RATE	NUMBER(4,2)	NOT NULL,
DATE_LAST_RAISE	DATE	NULL,

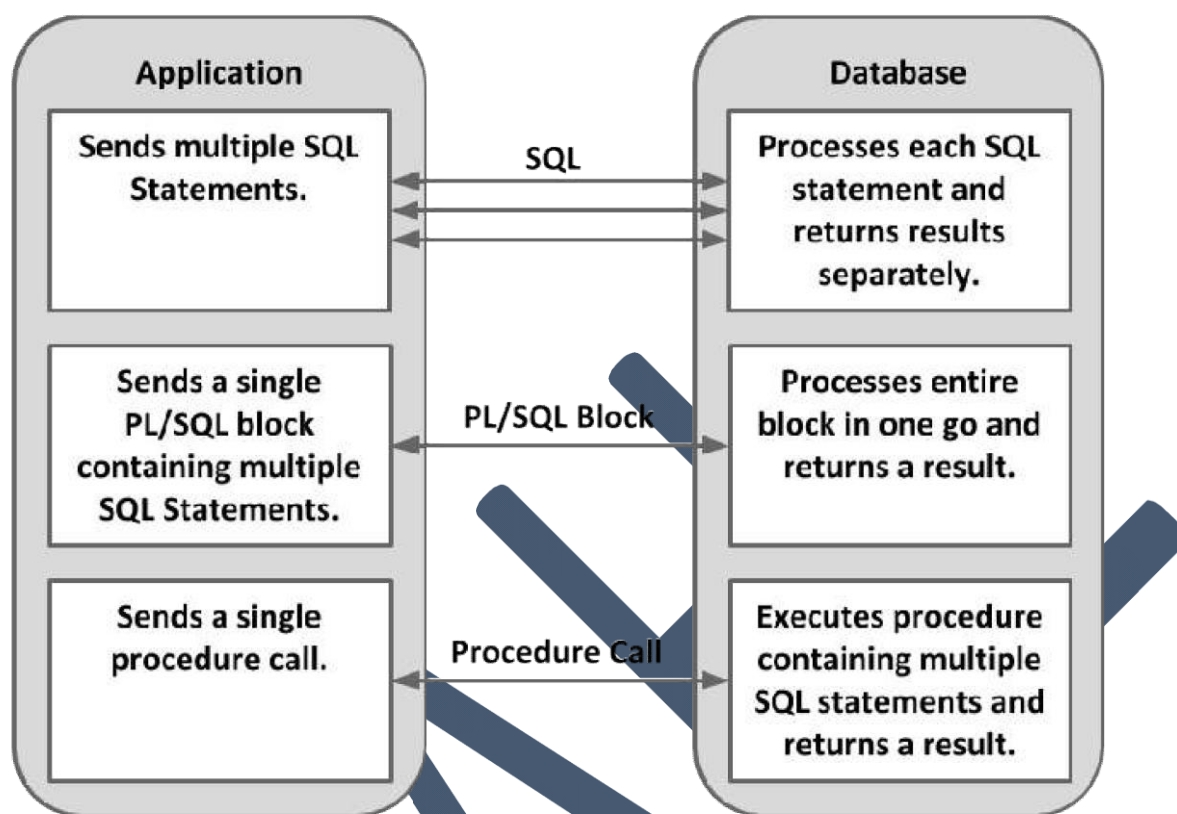
4. NOT NULL Constraints

Previous examples use the keywords NULL and NOT NULL listed on the same line as each column and after the data type. NOT NULL is a constraint that to place on a table's column. This constraint disallows the entrance of NULL values into a column; in other words, data is required in a NOT NULL column for each row of data in the table. NULL is generally the default for a column if NOT NULL is not specified, allowing NULL values in a column.

5. Check Constraints

Check (CHK) constraints can be utilized to check the validity of data entered into particular table columns. Check constraints are used to provide back-end database edits, although edits are commonly found in the front-end application as well. General edits restrict values that can be entered into columns or objects, whether within the database itself or on a front-end application. The check constraint is a way of providing another protective layer for the data.

```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID          CHAR(9)          NOT NULL,
EMP_NAME         VARCHAR2(40)     NOT NULL,
EMP_ST_ADDR      VARCHAR2(20)     NOT NULL,
EMP_CITY         VARCHAR2(15)     NOT NULL,
EMP_ST          CHAR(2)           NOT NULL,
EMP_ZIP          NUMBER(5)        NOT NULL,
EMP_PHONE        NUMBER(10)       NULL,
EMP_PAGER        NUMBER(10)       NULL),
PRIMARY KEY (EMP_ID),
CONSTRAINT CHK_EMP_ZIP CHECK ( EMP_ZIP = '46234');
```



Oracle is a multi-platform database, making PL/SQL and incredibly portable language.

Programming languages go in and out of fashion continually. Over the last 35+ years Oracle databases have remained part of the enterprise landscape. Suggesting that any language is a safer bet than PL/SQL is rather naive.

Centralizing application logic enables a higher degree of security and productivity. The use of Application Program Interfaces (APIs) can abstract complex data structures and security implementations from client application developers, leaving them free to do what they do best.

PL/SQL Architecture

CLASS: II B.COM CA
COURSE CODE: 16CCU402

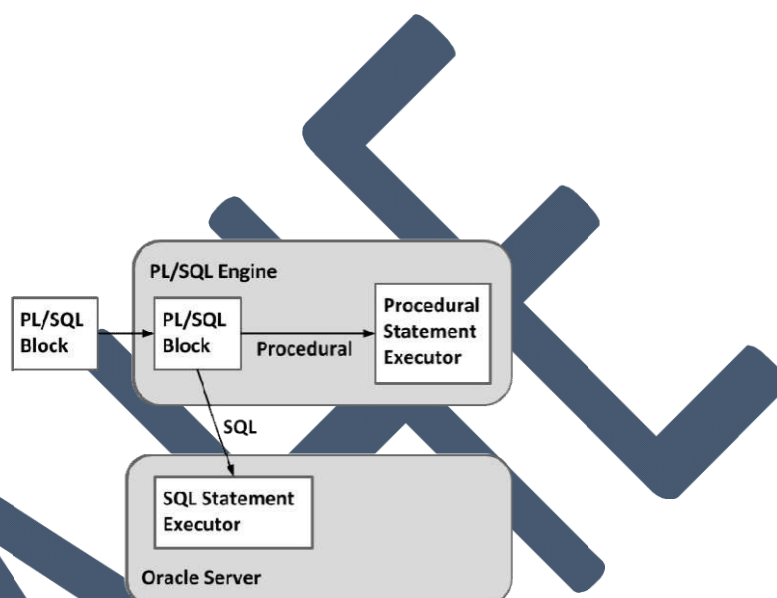
COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

The PL/SQL language is actually made up of two distinct languages. Procedural code is executed by the PL/SQL engine, while SQL is sent to the SQL statement executor.

The PL/SQL architecture mainly consists of following 3 components:

1. PL/SQL block
2. PL/SQL Engine
3. Database Server



PL/SQL block:

This is the component which has the actual PL/SQL code.

This consists of different sections to divide the code logically (declarative section for declaring purpose, execution section for processing statements, exception handling section for handling errors)

It also contains the SQL instruction that used to interact with the database server.

All the PL/SQL units are treated as PL/SQL blocks, and this is the starting stage of the architecture which serves as the primary input.

Following are the different type of PL/SQL units.

- Anonymous Block

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

- Function
- Library
- Procedure
- Package Body
- Package Specification
- Trigger
- Type
- Type Body

PL/SQL Engine

PL/SQL engine is the component where the actual processing of the codes takes place.

PL/SQL engine separates PL/SQL units and SQL part in the input (as shown in the image below).

The separated PL/SQL units will be handled with the PL/SQL engine itself.

The SQL part will be sent to database server where the actual interaction with database takes place.

It can be installed in both database server and in the application server.

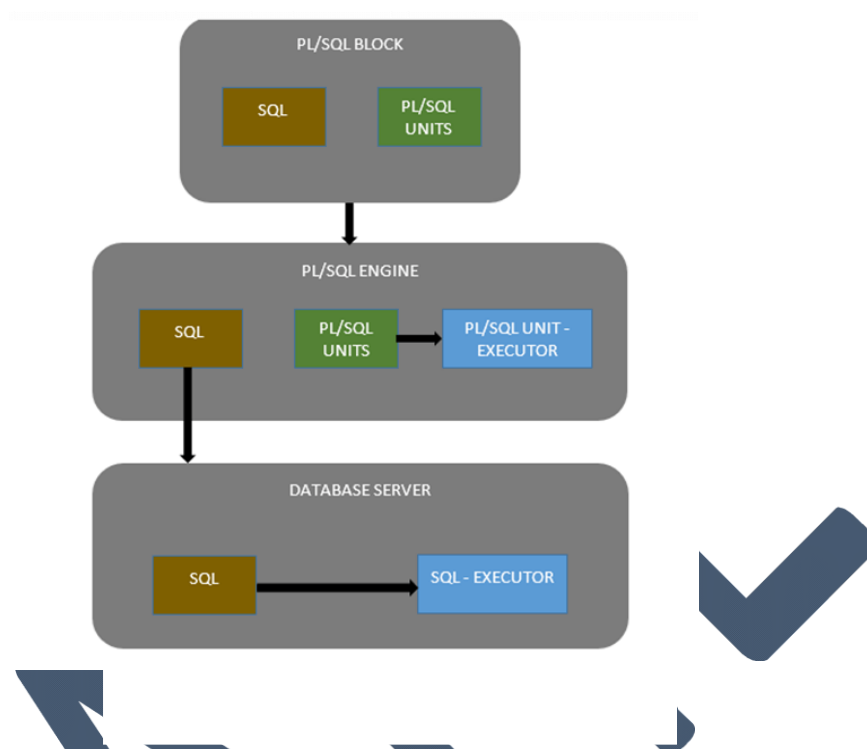
Database Server:

This is the most important component of PL/SQL unit which stores the data.

The PL/SQL engine uses the SQL from PL/SQL units to interact with the database server.

It consists of SQL executor which actually parses the input SQL statements and execute the same.

Below is the pictorial representation of Architecture of PL/SQL.



Features of PL/SQL

- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object-oriented programming.
- It supports the development of web applications and server pages.

Advantage of Using PL/SQL

- Better performance, as sql is executed in bulk rather than a single statement
- High Productivity

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

- Tight integration with SQL
- Full Portability
- Tight Security
- Support Object Oriented Programming concepts.

Basic Difference between SQL and PL/SQL

In this section, we will discuss some differences between SQL and PL/SQL

SQL	PL/SQL
<ul style="list-style-type: none"> SQL is a single query that is used to perform DML and DDL operations. 	<ul style="list-style-type: none"> PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc.
<ul style="list-style-type: none"> It is declarative, that defines what needs to be done, rather than how things need to be done. 	<ul style="list-style-type: none"> PL/SQL is procedural that defines how the things needs to be done.
<ul style="list-style-type: none"> Execute as a single statement. 	<ul style="list-style-type: none"> Execute as a whole block.
<ul style="list-style-type: none"> Mainly used to manipulate data. 	<ul style="list-style-type: none"> Mainly used to create an application.
<ul style="list-style-type: none"> Interaction with Database 	<ul style="list-style-type: none"> No interaction with the

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

server.	database server.
<ul style="list-style-type: none"> Cannot contain PL/SQL code in it. 	<ul style="list-style-type: none"> It is an extension of SQL, so it can contain SQL inside it.

Basic Syntax of PL/SQL

The Basic Syntax of PL/SQL which is a **block-structured** language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts –

S.No	Sections & Description
1	<p>Declarations</p> <p>This section starts with the keyword DECLARE. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.</p>
2	<p>Executable Commands</p> <p>This section is enclosed between the keywords BEGIN and END and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a NULL command to indicate that nothing should be executed.</p>

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Exception Handling

- 3 This section starts with the keyword **EXCEPTION**. This optional section contains **exception(s)** that handle errors in the program.

Every PL/SQL statement ends with a semicolon (;). PL/SQL blocks can be nested within other PL/SQL blocks using **BEGIN** and **END**.

Following is the basic structure of a PL/SQL block –

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling>

END;

Example

DECLARE

message varchar2(20):= 'Hello, World!';

BEGIN

dbms_output.put_line(message);

END;

/

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

The **end;** line signals the end of the PL/SQL block. To run the code from the SQL command line, we may need to type / at the beginning of the first blank line after the last line of the code. When the above code is executed at the SQL prompt, it produces the following result –

Hello World

PL/SQL procedure successfully completed.

The PL/SQL Identifiers

PL/SQL identifiers are constants, variables, exceptions, procedures, cursors, and reserved words. The identifiers consist of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs and should not exceed 30 characters.

By default, **identifiers are not case-sensitive.** So we can use **integer** or **INTEGER** to represent a numeric value. We cannot use a reserved keyword as an identifier.

The PL/SQL Delimiters

A delimiter is a symbol with a special meaning. Following is the list of delimiters in PL/SQL –

Delimiter	Description
+, -, *, /	Addition, subtraction/negation, multiplication, division
%	Attribute indicator
'	Character string delimiter

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

.	Component selector
(,)	Expression or list delimiter
:	Host variable indicator
,	Item separator
"	Quoted identifier delimiter
=	Relational operator
@	Remote access indicator
;	Statement terminator
:=	Assignment operator
=>	Association operator
	Concatenation operator
**	Exponentiation operator
<<, >>	Label delimiter (begin and end)
/*, */	Multi-line comment delimiter (begin and end)
--	Single-line comment indicator

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

..	Range operator
<, >, <=, >=	Relational operators
<>, !=, ~=, ^=	Different versions of NOT EQUAL

The PL/SQL Comments

Program comments are explanatory statements that can be included in the PL/SQL code that we write and helps anyone reading its source code. All programming languages allow some form of comments.

The PL/SQL supports single-line and multi-line comments. All characters available inside any comment are ignored by the PL/SQL compiler. The PL/SQL single-line comments start with the delimiter -- (double hyphen) and multi-line comments are enclosed by /* and */.

DECLARE

-- variable declaration

message varchar2(20):= 'Hello, World!';

BEGIN

/*

* PL/SQL executable statement(s)

*/

dbms_output.put_line(message);

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

END;

/

When the above code is executed at the SQL prompt, it produces the following result

—

Hello World

PL/SQL procedure successfully completed.

PL/SQL Program Units

A PL/SQL unit is any one of the following –

- PL/SQL block
- Function
- Package
- Package body
- Procedure
- Trigger
- Type
- Type body

Data Types

Data Types in PL/SQL.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

The PL/SQL variables, constants and parameters must have a valid data type, which specifies a storage format, constraints, and a valid range of values. We will focus on the **SCALAR** and the **LOB** data types.

S.No	Category & Description
1	Scalar Single values with no internal components, such as a NUMBER , DATE , or BOOLEAN .
2	Large Object (LOB) Pointers to large objects that are stored separately from other data items, such as text, graphic images, video clips, and sound waveforms.
3	Composite Data items that have internal components that can be accessed individually. For example, collections and records.
4	Reference Pointers to other data items.

PL/SQL Scalar Data Types and Subtypes

PL/SQL Scalar Data Types and Subtypes come under the following categories –

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

S.No	Date Type & Description
1	Numeric Numeric values on which arithmetic operations are performed.
2	Character Alphanumeric values that represent single characters or strings of characters.
3	Boolean Logical values on which logical operations are performed.
4	Datetime Dates and times.

PL/SQL provides subtypes of data types. For example, the data type NUMBER has a subtype called INTEGER. We can use the subtypes in our PL/SQL program to make the data types compatible with data types in other programs while embedding the PL/SQL code in another program, such as a Java program.

PL/SQL Numeric Data Types and Subtypes

Following table lists out the PL/SQL pre-defined numeric data types and their subtypes –

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

S.No	Data Type & Description
1	PLS_INTEGER Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits
2	BINARY_INTEGER Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits
3	BINARY_FLOAT Single-precision IEEE 754-format floating-point number
4	BINARY_DOUBLE Double-precision IEEE 754-format floating-point number
5	NUMBER(prec, scale) Fixed-point or floating-point number with absolute value in range 1E-130 to (but not including) 1.0E126. A NUMBER variable can also represent 0
6	DEC(prec, scale) ANSI specific fixed-point type with maximum precision of 38 decimal digits
7	DECIMAL(prec, scale)

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

	IBM specific fixed-point type with maximum precision of 38 decimal digits
8	NUMERIC(pre, scale) Floating type with maximum precision of 38 decimal digits
9	DOUBLE PRECISION ANSI specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)
10	FLOAT ANSI and IBM specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)
11	INT ANSI specific integer type with maximum precision of 38 decimal digits
12	INTEGER ANSI and IBM specific integer type with maximum precision of 38 decimal digits
13	SMALLINT ANSI and IBM specific integer type with maximum precision of 38 decimal digits

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

REAL

14

Floating-point type with maximum precision of 63 binary digits (approximately 18 decimal digits)

Following is a valid declaration –

DECLARE

num1 INTEGER;

num2 REAL;

num3 DOUBLE PRECISION;

BEGIN

null;

END;

/

When the above code is compiled and executed, it produces the following result –

PL/SQL procedure successfully completed

PL/SQL Character Data Types and Subtypes

Following is the detail of PL/SQL pre-defined character data types and their subtypes

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

S.No	Data Type & Description
1	CHAR Fixed-length character string with maximum size of 32,767 bytes
2	VARCHAR2 Variable-length character string with maximum size of 32,767 bytes
3	RAW Variable-length binary or byte string with maximum size of 32,767 bytes, not interpreted by PL/SQL
4	NCHAR Fixed-length national character string with maximum size of 32,767 bytes
5	NVARCHAR2 Variable-length national character string with maximum size of 32,767 bytes
6	LONG Variable-length character string with maximum size of 32,760 bytes
7	LONG RAW Variable-length binary or byte string with maximum size of 32,760 bytes, not interpreted by PL/SQL

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

8	ROWID Physical row identifier, the address of a row in an ordinary table
9	UROWID Universal row identifier (physical, logical, or foreign row identifier)

PL/SQL Boolean Data Types

The **BOOLEAN** data type stores logical values that are used in logical operations. The logical values are the Boolean values **TRUE** and **FALSE** and the value **NULL**.

However, SQL has no data type equivalent to **BOOLEAN**. Therefore, Boolean values cannot be used in –

- SQL statements
- Built-in SQL functions (such as **TO_CHAR**)
- PL/SQL functions invoked from SQL statements

PL/SQL Date time and Interval Types

The **DATE** datatype is used to store fixed-length datetimes, which include the time of day in seconds since midnight. Valid dates range from January 1, 4712 BC to December 31, 9999 AD.

The default date format is set by the Oracle initialization parameter **NLS_DATE_FORMAT**. For example, the default might be 'DD-MON-YY', which includes a two-digit number for the day of the month, an abbreviation of the month name, and the last two digits of the year. For example, 01-OCT-12.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Each DATE includes the century, year, month, day, hour, minute, and second. The following table shows the valid values for each field –

Field Name	Valid Datetime Values	Valid Interval Values
YEAR	-4712 to 9999 (excluding year 0)	Any nonzero integer
MONTH	01 to 12	0 to 11
DAY	01 to 31 (limited by the values of MONTH and YEAR, according to the rules of the calendar for the locale)	Any nonzero integer
HOUR	00 to 23	0 to 23
MINUTE	00 to 59	0 to 59
SECOND	00 to 59.9(n), where 9(n) is the precision of time fractional seconds	0 to 59.9(n), where 9(n) is the precision of interval fractional seconds
TIMEZONE_HOUR	-12 to 14 (range accommodates daylight savings time changes)	Not applicable

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

TIMEZONE_MINUTE	00 to 59	Not applicable
TIMEZONE_REGION	Found in the dynamic performance view V\$TIMEZONE_NAMES	Not applicable
TIMEZONE_ABBR	Found in the dynamic performance view V\$TIMEZONE_NAMES	Not applicable

PL/SQL Large Object (LOB) Data Types

Large Object (LOB) data types refer to large data items such as text, graphic images, video clips, and sound waveforms. LOB data types allow efficient, random, piecewise access to this data. Following are the predefined PL/SQL LOB data types –

Data Type	Description	Size
BFILE	Used to store large binary objects in operating system files outside the database.	System-dependent. Cannot exceed 4 gigabytes (GB).
BLOB	Used to store large binary objects in the database.	8 to 128 terabytes (TB)
CLOB	Used to store large blocks of character data in the database.	8 to 128 TB

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

NCLOB	Used to store large blocks of NCHAR data in the database.	8 to 128 TB
-------	---	-------------

PL/SQL User-Defined Subtypes

A subtype is a subset of another data type, which is called its base type. A subtype has the same valid operations as its base type, but only a subset of its valid values.

PL/SQL predefines several subtypes in package **STANDARD**. For example, PL/SQL predefines the subtypes **CHARACTER** and **INTEGER** as follows –

```
SUBTYPE CHARACTER IS CHAR;
```

```
SUBTYPE INTEGER IS NUMBER(38,0);
```

The following program illustrates defining and using a user-defined subtype

```
DECLARE
```

```
    SUBTYPE name IS char(20);
```

```
    SUBTYPE message IS varchar2(100);
```

```
    salutation name;
```

```
    greetings message;
```

```
BEGIN
```

```
    salutation := 'Reader ';
```

```
    greetings := 'Welcome to the World of PL/SQL';
```

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

```
dbms_output.put_line('Hello ' || salutation || greetings);
```

```
END;
```

When the above code is executed at the SQL prompt, it produces the following result

–

Hello Reader Welcome to the World of PL/SQL

PL/SQL procedure successfully completed.

NULLs in PL/SQL

PL/SQL NULL values represent **missing** or **unknown data** and they are not an integer, a character, or any other specific data type. Note that **NULL** is not the same as an empty data string or the null character value '\0'. A null can be assigned but it cannot be equated with anything, including itself.

Variable Declaration in PL/SQL

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

The syntax for declaring a variable is –

```
variable_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial_value]
```

Where, *variable_name* is a valid identifier in PL/SQL, *datatype* must be a valid

```
sales number(10, 2);
```

```
pi CONSTANT double precision := 3.1415;
```

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

```
name varchar2(25);
```

```
address varchar2(100);
```

When we provide a size, scale or precision limit with the data type, it is called a **constrained declaration**. Constrained declarations require less memory than unconstrained declarations. For example –

```
sales number(10, 2);
```

```
name varchar2(25);
```

```
address varchar2(100);
```

Initializing Variables in PL/SQL

Whenever we declare a variable, PL/SQL assigns it a default value of NULL. If we want to initialize a variable with a value other than the NULL value, we can do so during the declaration, using either of the following –

- The **DEFAULT** keyword
- The **assignment** operator

For example –

```
counter binary_integer := 0;
```

```
greetings varchar2(20) DEFAULT 'Have a Good Day';
```

We can also specify that a variable should not have a **NULL** value using the **NOT NULL** constraint. If we use the NOT NULL constraint, we must explicitly assign an initial value for that variable.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

It is a good programming practice to initialize variables properly otherwise, sometimes programs would produce unexpected results. Try the following example which makes use of various types of variables –

DECLARE

a integer := 10;

b integer := 20;

c integer;

f real;

BEGIN

c := a + b;

dbms_output.put_line('Value of c: ' || c);

f := 70.0/3.0;

dbms_output.put_line('Value of f: ' || f);

END;

/

When the above code is executed, it produces the following result –

Value of c: 30

Value of f: 23.333333333333333333

PL/SQL procedure successfully completed.

Variable Scope in PL/SQL

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

PL/SQL allows the nesting of blocks, i.e., each program block may contain another inner block. If a variable is declared within an inner block, it is not accessible to the outer block. However, if a variable is declared and accessible to an outer block, it is also accessible to all nested inner blocks. There are two types of variable scope –

- **Local variables** – Variables declared in an inner block and not accessible to outer blocks.
- **Global variables** – Variables declared in the outermost block or a package.

Following example shows the usage of **Local** and **Global** variables in its simple form

DECLARE

-- Global variables

num1 number := 95;

num2 number := 85;

BEGIN

dbms_output.put_line('Outer Variable num1: ' || num1);

dbms_output.put_line('Outer Variable num2: ' || num2);

DECLARE

-- Local variables

num1 number := 195;

num2 number := 185;

BEGIN

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

```
dbms_output.put_line('Inner Variable num1: ' || num1);
```

```
dbms_output.put_line('Inner Variable num2: ' || num2);
```

```
END;
```

```
END;
```

```
/
```

When the above code is executed, it produces the following result –

Outer Variable num1: 95

Outer Variable num2: 85

Inner Variable num1: 195

Inner Variable num2: 185

PL/SQL procedure successfully completed.

Assigning SQL Query Results to PL/SQL Variables

We can use the **SELECT INTO** statement of SQL to assign values to PL/SQL variables. For each item in the **SELECT list**, there must be a corresponding, type-compatible variable in the **INTO list**. The following example illustrates the concept.

Let us create a table named CUSTOMERS –

(For SQL statements, please refer to the [SQL tutorial](#))

```
CREATE TABLE CUSTOMERS(
```

```
    ID INT NOT NULL,
```

```
    NAME VARCHAR (20) NOT NULL,
```

```
    AGE INT NOT NULL,
```


CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

ADDRESS CHAR (25),

SALARY DECIMAL (18, 2),

PRIMARY KEY (ID)

);

Table Created

Let us now insert some values in the table –

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
```

```
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );
```

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
```

```
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );
```

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
```

```
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 ); INSERT INTO CUSTOMERS  
(ID,NAME,AGE,ADDRESS,SALARY)
```

```
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );
```

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
```

```
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );
```

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
```

```
VALUES (6, 'Komal', 22, 'MP', 4500.00 );
```

The following program assigns values from the above table to PL/SQL variables using the **SELECT INTO clause** of SQL –

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

DECLARE

```
c_id customers.id%type := 1;  
  
c_name customers.name%type;  
  
c_addr customers.address%type;  
  
c_sal customers.salary%type;
```

BEGIN

```
SELECT name, address, salary INTO c_name, c_addr, c_sal  
  
FROM customers  
  
WHERE id = c_id;  
  
dbms_output.put_line  
  
('Customer ' || c_name || ' from ' || c_addr || ' earns ' || c_sal);
```

END;

/

When the above code is executed, it produces the following result –

Customer Ramesh from Ahmedabad earns 2000

PL/SQL procedure completed successfully

Declaring a Constant

A constant is declared using the **CONSTANT** keyword. It requires an initial value and does not allow that value to be changed. For example –

```
PI CONSTANT NUMBER := 3.141592654;
```

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

DECLARE

-- constant declaration

pi constant number := 3.141592654;

-- other declarations

radius number(5,2);

dia number(5,2);

circumference number(7, 2);

area number (10, 2);

BEGIN

-- processing

radius := 9.5;

dia := radius * 2;

circumference := 2.0 * pi * radius;

area := pi * radius * radius;

-- output

dbms_output.put_line('Radius: ' || radius);

dbms_output.put_line('Diameter: ' || dia);

dbms_output.put_line('Circumference: ' || circumference);

dbms_output.put_line('Area: ' || area);

END;

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

/

When the above code is executed at the SQL prompt, it produces the following result

–

Radius: 9.5

Diameter: 19

Circumference: 59.69

Area: 283.53

PL/SQL procedure successfully completed.

The PL/SQL Literals

A literal is an explicit numeric, character, string, or Boolean value not represented by an identifier. For example, TRUE, 786, NULL, 'tutorialspoint' are all literals of type Boolean, number, or string. PL/SQL, literals are case-sensitive. PL/SQL supports the following kinds of literals –

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

The following table provides examples from all these categories of literal values.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

S.No	Literal Type & Example
1	Numeric Literals 050 78 -14 0 +32767 6.6667 0.0 -12.0 3.14159 +7800.00 6E5 1.0E-8 3.14159e0 -1E38 -9.5e-3
2	Character Literals 'A' '%' '9' ' ' 'z' '('
3	String Literals 'Hello, world!' 'Tutorials Point' '19-NOV-12'
4	BOOLEAN Literals TRUE, FALSE, and NULL.
5	Date and Time Literals DATE '1978-12-25'; TIMESTAMP '2012-10-29 12:01:01';

To embed single quotes within a string literal, place two single quotes next to each other as shown in the following program –

DECLARE

message varchar2(30):= 'That"s tutorialspoint.com!';

BEGIN

dbms_output.put_line(message);

END;

/

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

When the above code is executed at the SQL prompt, it produces the following result

PL/SQL procedure successfully completed.

Procedures in PL/SQL

A **subprogram** is a program unit/module that performs a particular task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the **calling program**.

A subprogram can be created –

- At the schema level
- Inside a package
- Inside a PL/SQL block

At the schema level, subprogram is a **standalone subprogram**. It is created with the CREATE PROCEDURE or the CREATE FUNCTION statement. It is stored in the database and can be deleted with the DROP PROCEDURE or DROP FUNCTION statement.

A subprogram created inside a package is a **packaged subprogram**. It is stored in the database and can be deleted only when the package is deleted with the DROP PACKAGE statement.

PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms –

- **Functions** – These subprograms return a single value; mainly used to compute and return a value.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

- **Procedures** – These subprograms do not return a value directly; mainly used to perform an action.

Parts of a PL/SQL Subprogram

Each PL/SQL subprogram has a name, and may also have a parameter list. Like anonymous PL/SQL blocks, the named blocks will also have the following three parts

S.No	Parts & Description
1	<p>Declarative Part</p> <p>It is an optional part. However, the declarative part for a subprogram does not start with the DECLARE keyword. It contains declarations of types, cursors, constants, variables, exceptions, and nested subprograms. These items are local to the subprogram and cease to exist when the subprogram completes execution.</p>
2	<p>Executable Part</p> <p>This is a mandatory part and contains statements that perform the designated action.</p>
3	<p>Exception-handling</p> <p>This is again an optional part. It contains the code that handles run-time errors.</p>

Creating a Procedure

A procedure is created with the **CREATE OR REPLACE PROCEDURE** statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows

–

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

CREATE [OR REPLACE] PROCEDURE procedure_name

[(parameter_name [IN | OUT | IN OUT] type [, ...])]

{IS | AS}

BEGIN

< procedure_body >

END procedure_name;

Where,

- *procedure-name* specifies the name of the procedure.
- [OR REPLACE] option allows the modification of an existing procedure.
- The optional parameter list contains name, mode and types of the parameters. **IN** represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
- *procedure-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone procedure.

Example

The following example creates a simple procedure that displays the string 'Hello World!' on the screen when executed.

CREATE OR REPLACE PROCEDURE greetings

AS

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

BEGIN

```
dbms_output.put_line('Hello World!');
```

END;

/

When the above code is executed using the SQL prompt, it will produce the following result –

Procedure created.

Executing a Standalone Procedure

A standalone procedure can be called in two ways –

- Using the **EXECUTE** keyword
- Calling the name of the procedure from a PL/SQL block

The above procedure named '**greetings**' can be called with the EXECUTE keyword as –

```
EXECUTE greetings;
```

The above call will display –

Hello World

PL/SQL procedure successfully completed.

The procedure can also be called from another PL/SQL block –

BEGIN

```
greetings;
```

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

END;

/

The above call will display –

Hello World

PL/SQL procedure successfully completed.

Deleting a Standalone Procedure

A standalone procedure is deleted with the **DROP PROCEDURE** statement. Syntax for deleting a procedure is –

DROP PROCEDURE procedure-name;

DROP PROCEDURE greetings;

Parameter Modes in PL/SQL Subprograms

The following table lists out the parameter modes in PL/SQL subprograms –

S.No	Parameter Mode & Description
1	IN An IN parameter , to pass a value to the subprogram. It is a read-only parameter. Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. We can pass a constant, literal, initialized variable, or expression as an IN parameter. We can also initialize it to a default value; however, in that case, it is omitted from the subprogram call. It is the default mode of parameter passing. Parameters are passed by reference.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

2	<p>OUT</p> <p>An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. We can change its value and reference the value after assigning it. The actual parameter must be variable and it is passed by value.</p>
3	<p>IN OUT</p> <p>An IN OUT parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and the value can be read.</p> <p>The actual parameter corresponding to an IN OUT formal parameter must be a variable, not a constant or an expression. Formal parameter must be assigned a value. Actual parameter is passed by value.</p>

IN & OUT Mode Example 1

This program finds the minimum of two values. Here, the procedure takes two numbers using the IN mode and returns their minimum using the OUT parameters.

DECLARE

a number;

b number;

c number;

PROCEDURE findMin(x IN number, y IN number, z OUT number) IS

BEGIN

IF x < y THEN

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

```
z:= x;  
  
ELSE  
  
z:= y;  
  
END IF;  
  
END;  
  
BEGIN  
  
a:= 23;  
  
b:= 45;  
  
findMin(a, b, c);  
  
dbms_output.put_line(' Minimum of (23, 45) : ' || c);  
  
END;  
  
/
```

When the above code is executed at the SQL prompt, it produces the following result

–

Minimum of (23, 45) : 23

PL/SQL procedure successfully completed.

IN & OUT Mode Example 2

This procedure computes the square of value of a passed value. This example shows how we can use the same parameter to accept a value and then return another result.

DECLARE

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

a number;

PROCEDURE squareNum(x IN OUT number) IS

BEGIN

x := x * x;

END;

BEGIN

a:= 23;

squareNum(a);

dbms_output.put_line(' Square of (23): ' || a);

END;

/

When the above code is executed at the SQL prompt, it produces the following result

-

Square of (23): 529

PL/SQL procedure successfully completed.

Methods for Passing Parameters

Actual parameters can be passed in three ways -

- Positional notation
- Named notation
- Mixed notation

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Positional Notation

In positional notation, we can call the procedure as –

```
findMin(a, b, c, d);
```

In positional notation, the first actual parameter is substituted for the first formal parameter; the second actual parameter is substituted for the second formal parameter, and so on. So, **a** is substituted for **x**, **b** is substituted for **y**, **c** is substituted for **z** and **d** is substituted for **m**.

Named Notation

In named notation, the actual parameter is associated with the formal parameter using the **arrow symbol (=>)**. The procedure call will be like the following –

```
findMin(x => a, y => b, z => c, m => d);
```

Mixed Notation

In mixed notation, we can mix both notations in procedure call; however, the positional notation should precede the named notation.

The following call is legal –

```
findMin(a, b, c, m => d);
```

However, this is not legal:

```
findMin(x => a, b, c, d);
```

Functions in PL/SQL

A function is same as a procedure except that it returns a value.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Creating a Function

A standalone function is created using the CREATE FUNCTION statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows

```
CREATE [OR REPLACE] FUNCTION function_name
```

```
[(parameter_name [IN | OUT | IN OUT] type [, ...])] 
```

```
RETURN return_datatype
```

```
{IS | AS}
```

```
BEGIN
```

```
< function_body >
```

```
END [function_name];
```

Where,

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

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Example

The following example illustrates how to create and call a standalone function. This function returns the total number of CUSTOMERS in the customers table.

To use the CUSTOMERS table, which we had created in the [PL/SQL Variables](#) we –

Select * from customers;

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	hmedabad	000.00
2	Khilan	25	Delhi	500.00
3	kaushik	23	Kota	000.00
4	Chaitali	25	Mumbai	500.00
5	Hardik	27	Bhopal	500.00
6	Komal	22	MP	500.00

CREATE OR REPLACE FUNCTION totalCustomers

RETURN number IS

total number(2) := 0;

BEGIN

SELECT count(*) into total

FROM customers;

RETURN total;

END;

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

/

When the above code is executed using the SQL prompt, it will produce the following result –

Function created.

Calling a Function

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

A called function performs the defined task and when its return statement is executed or when the last end statement is reached, it returns the program control back to the main program.

To call a function, simply need to pass the required parameters along with the function name and if the function returns a value, then we can store the returned value. Following program calls the function totalCustomers from an anonymous block –

```
DECLARE  
  
    c number(2);  
  
BEGIN  
  
    c := totalCustomers();  
  
    dbms_output.put_line('Total no. of Customers: ' || c);  
  
END;  
  
/
```

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

When the above code is executed at the SQL prompt, it produces the following result

–

Total no. of Customers: 6

PL/SQL procedure successfully completed.

Example

The following example demonstrates Declaring, Defining, and Invoking a Simple PL/SQL Function that computes and returns the maximum of two values.

DECLARE

a number;

b number;

c number;

FUNCTION findMax(x IN number, y IN number)

RETURN number

IS

z number;

BEGIN

IF x > y THEN

z:= x;

ELSE

Z:= y;

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

END IF;

RETURN z;

END;

BEGIN

a:= 23;

b:= 45;

c := findMax(a, b);

dbms_output.put_line(' Maximum of (23,45): ' || c);

END;

/

When the above code is executed at the SQL prompt, it produces the following result

–

Maximum of (23,45): 45

PL/SQL procedure successfully completed.

PL/SQL Recursive Functions

A program or subprogram may call another subprogram. When a subprogram calls itself, it is referred to as a recursive call and the process is known as recursion.

To illustrate the concept, let us calculate the factorial of a number. Factorial of a number n is defined as –

$$n! = n*(n-1)!$$

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

$$= n*(n-1)*(n-2)!$$

...

$$= n*(n-1)*(n-2)*(n-3)... 1$$

The following program calculates the factorial of a given number by calling itself recursively -

DECLARE

num number;

factorial number;

FUNCTION fact(x number)

RETURN number

IS

f number;

BEGIN

IF x=0 THEN

f := 1;

ELSE

f := x * fact(x-1);

END IF;

RETURN f;

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

END;

BEGIN

num:= 6;

factorial := fact(num);

dbms_output.put_line(' Factorial ' || num || ' is ' || factorial);

END;

/

When the above code is executed at the SQL prompt, it produces the following result

-

Factorial 6 is 720

PL/SQL procedure successfully completed.

Cursor Management

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.

To a cursor name 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, to 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.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

3	<p>%ISOPEN</p> <p>Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.</p>
4	<p>%ROWCOUNT</p> <p>Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.</p>

Any SQL cursor attribute will be accessed as `sql%attribute_name` as shown below in the example.

Example

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

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

```
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 we check the records in customers table, to 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

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

3	kaushik	23	Kota	2500.00
4	Chaitali	25	Mumbai	7000.00
5	Hardik	27	Bhopal	9000.00
6	Komal	22	MP	5000.00

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the context area. An explicit cursor 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.

The syntax for creating an explicit cursor is –

CURSOR cursor_name IS select_statement;

Working with an explicit cursor includes the following steps –

- Declaring the cursor for initializing the memory
- Opening the cursor for allocating the memory
- Fetching the cursor for retrieving the data
- Closing the cursor to release the allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example –

CURSOR c_customers IS

SELECT id, name, address FROM customers;

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows –

```
OPEN c_customers;
```

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows –

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows –

```
CLOSE c_customers;
```

Example

Following is a complete example to illustrate the concepts of explicit cursors &minua;

```
DECLARE
```

```
    c_id customers.id%type;
```

```
    c_name customerS.No.ame%type;
```

```
    c_addr customers.address%type;
```

```
    CURSOR c_customers is
```

```
        SELECT id, name, address FROM customers;
```

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

BEGIN

OPEN c_customers;

LOOP

FETCH c_customers into c_id, c_name, c_addr;

EXIT WHEN c_customers%notfound;

dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);

END LOOP;

CLOSE c_customers;

END;

/

When the above code is executed at the SQL prompt, it produces the following result

–

1 Ramesh Ahmedabad

2 Khilan Delhi

3 kaushik Kota

4 Chaitali Mumbai

5 Hardik Bhopal

6 Komal MP

PL/SQL procedure successfully completed.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Exceptions in PL/SQL

An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using **EXCEPTION** block in the program and an appropriate action is taken against the error condition. There are two types of exceptions –

- System-defined exceptions
- User-defined exceptions

Syntax for Exception Handling

The general syntax for exception handling is as follows. The default exception will be handled using **WHEN others THEN** –

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling goes here >

WHEN exception1 THEN

exception1-handling-statements

WHEN exception2 THEN

exception2-handling-statements

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

WHEN exception3 THEN

exception3-handling-statements

.....

WHEN others THEN

exception3-handling-statements

END;

Example

using the CUSTOMERS table

DECLARE

c_id customers.id%type := 8;

c_name customerS.No.ame%type;

c_addr customers.address%type;

BEGIN

SELECT name, address INTO c_name, c_addr

FROM customers

WHERE id = c_id;

DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);

DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);

EXCEPTION

WHEN no_data_found THEN

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

```
dbms_output.put_line('No such customer!');
```

```
WHEN others THEN
```

```
dbms_output.put_line('Error!');
```

```
END;
```

```
/
```

When the above code is executed at the SQL prompt, it produces the following result

–

No such customer!

PL/SQL procedure successfully completed.

The above program displays the name and address of a customer whose ID is given. Since there is no customer with ID value 8 in our database, the program raises the run-time exception **NO_DATA_FOUND**, which is captured in the **EXCEPTION block**.

Raising Exceptions

Exceptions are raised by the database server automatically whenever there is any internal database error, but exceptions can be raised explicitly by the programmer by using the command **RAISE**. Following is the simple syntax for raising an exception –

```
DECLARE
```

```
exception_name EXCEPTION;
```

```
BEGIN
```

```
IF condition THEN
```

```
RAISE exception_name;
```

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

END IF;

EXCEPTION

WHEN exception_name THEN

statement;

END;

User-defined Exceptions

PL/SQL allows to define own exceptions according to the need of program. A user-defined exception must be declared and then raised explicitly, using either a RAISE statement or the procedure **DBMS_STANDARD.RAISE_APPLICATION_ERROR**.

The syntax for declaring an exception is –

DECLARE

my-exception EXCEPTION;

Example

The following example illustrates the concept. This program asks for a customer ID, when the user enters an invalid ID, the exception **invalid_id** is raised.

DECLARE

c_id customers.id%type := &cc_id;

c_name customerS.No.ame%type;

c_addr customers.address%type;

-- user defined exception

ex_invalid_id EXCEPTION;

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

BEGIN

IF c_id <= 0 THEN

RAISE ex_invalid_id;

ELSE

SELECT name, address INTO c_name, c_addr

FROM customers

WHERE id = c_id;

DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);

DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);

END IF;

EXCEPTION

WHEN ex_invalid_id THEN

dbms_output.put_line('ID must be greater than zero!');

WHEN no_data_found THEN

dbms_output.put_line('No such customer!');

WHEN others THEN

dbms_output.put_line('Error!');

END;

/

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

When the above code is executed at the SQL prompt, it produces the following result

–

Enter value for cc_id: -6 (let's enter a value -6)

old 2: c_id customers.id%type := &cc_id;

new 2: c_id customers.id%type := -6;

ID must be greater than zero!

PL/SQL procedure successfully completed.

Pre-defined Exceptions

PL/SQL provides many pre-defined exceptions, which are executed when any database rule is violated by a program. For example, the predefined exception NO_DATA_FOUND is raised when a SELECT INTO statement returns no rows. The following table lists few of the important pre-defined exceptions –

Exception	Oracle Error	SQLCODE	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 clause of a CASE

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

			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.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

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 a 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

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

			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 sizeconstraint error occurs.
ZERO_DIVIDE	01476	1476	It is raised when an attempt is made to divide a

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

			number by zero.
--	--	--	-----------------

Triggers in PL/SQL

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

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

- 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,

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

- **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 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

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

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

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.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

The following points need to be considered here –

- OLD and NEW references are not available for table-level triggers, rather to can use them for record-level triggers.
- If we want to query the table in the same trigger, then we 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.
- The above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but we can write trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using the DELETE operation on the table.

Triggering a Trigger

DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table –

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (7, 'Kriti', 22, 'HP', 7500.00 );
```

When a record is created in the CUSTOMERS table, the above create trigger, display_salary_changes will be fired and it will display the following result –

Old salary:

New salary: 7500

Salary difference:

Because this is a new record, old salary is not available and the above result comes as null. Let us now perform one more DML operation on the CUSTOMERS table. The UPDATE statement will update an existing record in the table –

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

UPDATE customers

SET salary = salary + 500

WHERE id = 2;

When a record is updated in the CUSTOMERS table, the above create trigger, display_salary_changes will be fired and it will display the following result

Old salary: 1500

New salary: 2000

Salary difference: 500

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

POSSIBLE QUESTION

Part A (1Mark)

(Online Examinations)

Part B (2 Marks)

1. How are exceptions handled in PL/SQL?
2. What are the different types of cursors.
3. What is DDL, DCL, and DML (Data Manipulation Language)?
4. What are referential integrity constraints?
5. What is the need for triggers?
6. Give the forms of triggers
7. What is a cursor?
8. What types of cursors are supported in PL/SQL?
9. What is the purpose of cursor in PL/SQL?
10. What is trigger?
11. Write the syntax for defining a trigger.
12. What are the uses of using triggers?
13. Define: functions
14. What are the Exceptions in PL/SQL?
15. Mention the categories of SQL.

CLASS: II B.COM CA
COURSE CODE: 16CCU402

COURSE NAME: DATABASE MANAGEMENT SYSTEM
UNIT: V (ORACLE)

BATCH: 2016-2019

Part C (6 Marks)

1. Explain Triggers and its types with examples.
2. Illustrate a block of PL/SQL with neat diagram.
3. What is a cursor? What types of cursors are supported in PL/SQL? Explain Cursor attributes.
4. Write a PL/SQL code to display the Empno, Ename and Job of employees of DeptNo 10 with CURSOR FOR LOOP Statement.
5. What is the importance of cursor for loop? How it simplifies the operation? Explain with suitable example.
Explain Triggers and its types with examples.
6. What is cursor? What are the cursors attributes? Explain.
7. Explain control structure statement on PL/SQL block.
8. Differentiate PL/SQL and SQL?
9. What are the data types available in PL/SQL? Explain it.
10. Explain types of cursor?



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-641021
(For candidates admitted from 2017 onwards)

SUBJECT NAME: DATABASE MANAGEMENT SYSTEM

SEMESTER: IV

SUBJECT CODE:17CCU402

UNIT- V

S. NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWERS
1	A query has a _____ syntax	Dimensional	Two dimensional	No syntax	Three dimensional	Two dimensional
2	The _____ is used to set for the negative value.	Sys_code	Sql	Select	Sis_code	Sys_code
3	A trigger can be dropped by using _____ command	Drop trigger	Alter trigger drop	Define drop trigger	Declare drop trigger	Drop trigger
4	The variable in the trigger are declared by using _____ symbol	+=	@	/	/@	@
5	The _____ is which holds another object.	Procedure	Function	Package	Trigger	Package
6	The subprogram created inside a package is called _____	Combined subprogram	Collective subprogram	Nested subprogram	Packaged subprogram	Packaged subprogram
7	The _____ is called private work area	Triggers	Packages	Cursors	Packages	Cursors

8	The data that is stored in the cursor is called the _____	Active dataset	Packages	Relations	Tables	Active dataset
9	A _____ deals with the handling of errors that arise which make up the sql code block.	Begin	Exception	End	Declare	Exception
10	A _____ cursor is automatically declared by oracle every time a sql statement is executed.	Implicit cursor	Explicit cursor	Internal cursor	External cursor	Implicit cursor
11	The _____ returns a value back to the calling block.	Procedure	Package	Function	Trigger	Function
12	The execution of a trigger is also known as _____	Executing the trigger	Firing the trigger	Perform the trigger	Implement trigger	Firing the trigger
13	Which keyword is used to delete the trigger?	Delete	Drop	Remove	Truncate	Drop
14	The objects in the specification section of a package are called _____	Private objects	Public objects	Package variable	Modules header	Public objects
15	The _____ trigger is fired before execution of a DML statement.	Previous	After	Before	Prior	Before
16	A each row trigger is known as _____	Trigger	Tuple trigger	Record trigger	Row trigger	Row trigger

17	The pressing _____ Key used for a skeleton of the table	Function	Shift+insert	Ctrl+shift	Alt+insert	Function
18	The QBE stands for _____	Query by example	Query by equation	Query by embedded	Query be embedde d	Query by example
19	The PL/I stands for _____	Programming language	Physical database record	Physical database language	Physical language	Programming language
20	An oracle was developed in the year _____	1977	1978	1979	1980	1977
21	In oracle _____ data type used for strings	Char	Varchar	Fixed bin	Date	Varchar
22	The varchar data type can accept _____ bytes	2000	4000	1000	3000	4000
23	A char data type can accept _____ bytes	2000	4000	1000	5000	2000
24	The initial release of an oracle is in the year _____	1977	1978	1979	1980	1979

25	The ____ action used in an explicit cursor processes the query and creates an active set of rows available in the cursor	Declare	Open	Fetch	Close	Fetch
26	Which keyword is used to specify the trigger restriction?	control	restrict	check	when	when
27	A _____ is the building block of modular programming.	Packages	Procedure	Cursor	Exception	Procedure
28	The Oracle can be used as _____ for projects	Backend	frontend	Middle	Programming language	Back end
29	A _____ datatype trims the unwanted space in the memory	Char	Varchar	Fixed bin	Date	Varchar
30	The _____ trigger fires after a DML statement is executed.	After	Later	Next	When	After
31	The _____ command is used to delete the whole table in the database	Drop	Delete	Drop and delete	Alter	Drop
32	Which of the following is DDL command?	Insert	Delete	Insert and delete	Create	Create
33	Which of the following is DML command?	Alter	Drop	Insert	Create	Insert
34	PL/SQL is a superset of _____	Sql	Pl/Sql	Linux	Unix	Sql

35	The sql statements are passed to the oracle engine__	One at a time	Twice a time	Three times	Unlimited	One at a time
36	The_____key is used to errors occurred in sql.	backspace	delete	insert	alt	delete
37	The _____ can be a procedure or function	stored procedure	query	nested program	subprogram	subprogram
38	The major commands in sql_____	Create	Control	Enter	Exit	Create
39	The actions to perform on a database are done with_____	SQL statements	functions	COBOL	BASIC	SQL statements
40	The _____ statement disables the cursor	Disable cursor	Close cursor	End cursor	Remove cursor	Close cursor
41	The sql keywords are written in_____	Uppercase	Lowercase	Uppercase and Lowercase	Special characters	Uppercase
42	The_____ command is used to modify a table _____	Alter	Update	Delete	Drop	Alter

43	The _____ command is used to create a new database	Update	Create	Insert	Select	Create
44	The__ command is used to create a new table	Update	Create table	Insert	Select	Create table
45	The _____ command is used to extracts data from a database	Select	Create table	Select and create table	Where	Select
46	The _____ command is used to updates the data in a database	Update	Alter	Update and alter	Where	Update
47	The_____ command is used to delete a data from a database	Delete	Update	Insert	Where	Delete
48	The_____command is used to insert new data into a database	Insert into	Update	Insert	Where	Insert into
49	The _____command is used to delete a table.	Drop Table	Select	Update	Where	Drop table
50	The _____clause is used to extract only those records that fulfill a specified condition	Where	SELECT	And	Update	Where

51	The__ clause is used in a select statement to sort results either in ascending or descending order.	Order by	Sql Functions	Rollback	Commit	Order by
52	The__ operator is used to select values within a range	Between	And	Or	Where	Between
53	The_____ command is used to temporarily rename a table or a column heading	Commit	Aliases	Roll Back	Select	Aliases
54	The_____function returns the number of rows that matched a specified criteria	Count()	Min()	Avg()	Sum()	Count()

55	The_____function returns the smallest value of the selected column	Min()	Max()	Min() and max()	Avg()	Min()
56	The_____ function returns the largest values of the selected column	Max()	Min()	Min() and sum()	Sum()	Max()
57	The_____ function returns the average value of a numeric column	Avg()	Max()	Avg() and Sum()	Sum()	Avg()
58	The _____function returns the current system date and time.	Last()	Now()	First ()	Sum()	Now()
59	The_____ function returns the last value of the selected column.	Last()	Max()	Avg()	Now()	Last()
60	The _____function returns the first value of the selected column.	Last()	Max()	First()	Sum()	First()

Reg.....
[17CCU402]

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-641021
(For candidates admitted from 2017 onwards)
CONTINUOUS INTERNAL ASSESSMENT III, March 2019
BACHELOR OF COMMERCE (COMPUTER APPLICATION)
FOURTH SEMESTER
DATABASE MANAGEMENT SYSTEM

Time: 2 Hours
Date: 12.3.19

Maximum: 50 Marks
Session : FN

PART – A (20 X 1 = 20 Marks)
ANSWER ALL THE QUESTIONS

1. The first normal form is used to_____.
 - a) Avoids repetition
 - b) Remove repetition
 - c) Delete repetition
 - d) Accept repetition
2. The FD stands for_____.
 - a) Functional dependency
 - b) Facilitate dependency
 - c) Functional data
 - d) Facilitate independence
3. The _____is a group of dependencies.
 - a) Facilitate
 - b) Non-functional
 - c) Trivial functional
 - d) Normalization
4. The BCNF stands for_____.
 - a) Boyce-Codd Normal Forms
 - b) Boce-Codd Normal Forms
 - c) Boyce- C Normal Forms
 - d) B- Codd Normal Forms
5. The _____is a bottom-up approach to design a database.
 - a) Functional dependency
 - b) Database modelling
 - c) Normalization
 - d) Decomposition
6. A relational database, the functional dependency between two attributes A and B is denoted by_____relation.
 - a) $A \rightarrow B$
 - b) $B \leftarrow A$
 - c) $AB \rightarrow R$
 - d) $R \leftarrow AB$
7. Which forms are based on the concept of functional Dependency?
 - a) First normal form
 - b) Second normal form

- c) Third normal form d) Fourth normal form
8. The constraint between two different attributes sets is classified as _____.
 a) Modification anomaly b) Functional dependency
 c) Insertion anomaly d) Deletion anomaly
9. The normalization form which is based on the transitive dependency is classified as _____.
 a) First normal form b) Second normal form
 c) Fourth normal form d) Third normal form
10. The _____ normal form only includes single atomic values.
 a) Third b) First
 c) Second d) Fourth
11. The _____ is used to set for the negative value.
 a) Sys_code b) Sql c) Select d) Sis_code
12. A trigger can be dropped by using _____ command.
 a) Drop trigger b) Alter trigger drop
 c) Define drop trigger d) Declare drop trigger
13. The variable in the triggers is declared by using _____ symbol.
 a) += b) @ c) / d) /@
14. The subprogram created inside a package is called _____.
 a) Combined subprogram b) Collective subprogram
 c) Nested subprogram d) Packaged subprogram
15. The _____ is called private work area.
 a) Trigger b) Package c) Cursor d) Package
16. A _____ deals with the handling of errors that arise which make up the sql code block.
 a) Begin b) Exception c) End d) Declare
17. A _____ cursor is automatically declared by oracle every time a sql statement is executed.
 a) Implicit cursor b) Explicit cursor c) Internal cursor d) External cursor
18. The execution of a trigger is also known as _____.
 a) Executing the trigger b) Firing the trigger
 c) Perform the trigger d) Implement trigger
19. The objects in the specification section of packages are called _____.

a) Private objects

b) Public objects

c) Package variable

d) Modules header

20. Which keyword is used to delete the trigger?

a) Delete

b) Drop

c) Remove

d) Truncate

PART - B (3 X 2 = 6 MARKS)

(ANSWER ALL THE QUESTIONS)

21. Why the fourth normal form is more desirable than BCNF?

22. Compare and contrast the lossless decomposition and dependency preservation.

23. What are the problems caused by redundancy?

PART - C (3 X 8 = 24 MARKS)

(ANSWER ALL THE QUESTIONS)

24 a. Explain the various types of normal forms with an example.

(OR)

b. Give a relation R with four attributes $R = \{A\ B\ C\ D\}$ and the following Functional dependency, identify the candidate keys for R and the highest normal forms.

25 a. Illustrate the various desirable properties of decomposition with an appropriate example.

(OR)

b. Write the DDL, DML commands for the student database which contains student details: name, id, DOB, branch, DOJ. Course details, Course name, Course id, Stud. id, Faculty name, id, marks.

26 a. Discuss in detail about simple cursor loops with its syntax and an example in PL/SQL block.

(OR)

b. How is the exception section handled in PL/SQL? Give the syntax and write a PL/SQL program to print the student mark percentage.

* * * * *