Database Management Systems

Chapter 1 : Introduction Database Concepts

Q. 1 Discuss Database system.

Ans. : Database Management System (DBMS) May 2017

A **Database Management System (DBMS)** is a collection of software or programs which help user in creation and maintenance of a database (set of information). Hence it is also known as a computerized record-keeping system. DBMS is the software system that helps in the process of defining, constructing, manipulating the database. Database management system has become an integral part of the information systems of many organizations as it is used to handle a huge amount of data.

Computer-based **Information Systems (IS)** is capable of serving to many complex tasks in a coordinated manner. Such systems handle large volumes of data, multiple users and several applications in a centralized database environment. The heart of an Information System (IS) is database management system. This is because most **Information Systems (IS)** have to handle huge amounts of data. This core module of an IS is also called as Database Management System (DBMS).

Examples

1. MS Access, Fox Pro by Microsoft.
2. Oracle by Oracle corp.
3. SQL Server By Microsoft.
4. Ingres, DB2 by IBM.

Q. 2 Explain the features of DBMS.

OR Explain various advantages of Databases.

Ans. : Characteristics of DBMS

The database approach has many important characteristics due to which database has become an integral part of the software industry. **The various characteristics of the databases are as mentioned below :**

1. **Data integrity :** Integrity constraints provide a way of ensuring that changes made to the database by authorized users that do not result in a loss of data consistency and correctness. Database integrity concern with the correctness and completeness of data in the database. This objective can never be guaranteed, one cannot ensure that every entry made in database is accurate.

   Some examples of incorrect data are as below :

   1. Student taking admission to branch which is not available in college.
   2. Employee assigned with non existing department.
   3. Sometime inconsistency introduced due to system failures.

2. **Data security :** A DBMS system always has a separate system for security which is responsible for protecting database against accidental or intentional loss, destruction or misuse. Data in database should be given to only authorized users. Only authorized users should be allowed to modify data. Authorized users are able to access data any time he wants.
3. **Data Independence :** Data Independence can be defined as the capacity to change data kept at one place without changing data kept at other locations.
4. **Transaction control - rollback :** The changes made to database can be reverted back with help of rollback command. The changes can be saved successful with help of commit data command.
5. **Concurrency control :** The data in database can be accessed by multiple users at same point of time. Such operations allowed by sharing same data between multiple users.
6. **Data recovery - backup and restore :** Database recovery is the process of restoring the database to original (correct) state after database failure. The main element of database recovery is the most recent database backup. If maintain database backup efficiently, then database recovery is very straight forward process.

Q. 3 Explain advantages of DBMS over file system.

OR List four significant differences between file processing system and database management system.

May 2012, Dec. 2014, May 2015, Dec. 2015

Ans. : File System v/s Database System

1. **Redundancy can be reduced :** As using relational approach for data organization, data is not stored in more than one location. Repetition of information can be avoided which in turn saves storage space.
2. **Inconsistency can be avoided :** With the usage of database, it is assured that all the users access actual or true data present in the database.
3. **Data can be shared :** Multiple users can login at a time into the database to access information. They can manipulate the database in a controlled environment.

   Example : In yahoo portal, many users are accessing data in database in a controlled manner.
4. **With a centralized control of data,** the database system may be designed for an overall optimal performance for entire organization.
5. **Standards can be enforced :** Standards (rules and regulations for coding and designing) can be enforced on the database to regulate the access to the database. Primary Key constraint or foreign key constraint can be enforced on database which will be helpful for accessing data from database.
6. **Security restrictions can be applied :** Security is the process of limiting access to the database server itself for some users.

It is the most important for security and needs to be carefully planned.

7. **Integrity can be maintained :** Through integrity, one can ensure only accurate data is stored within the database.
8. **Data independence can be provided :** None of the users need to know the technical aspects of the database to access it. They are physically as well as logically independent to access the database.
9. **New applications** may be developed using the existing database.

Q. 4 Discuss different Database Users. May 2017

Ans. : Database Users

1. **Naive users :** Naive users are users who interact with the system using application programs that have been developed previously. For example, Student wants to pay fees Rs.50 then accountant will invokes a program called fees_payment(). This program asks the accountant for the amount of fees to be paid.

   The typical graphical user interface for naive users is a kind of form interface, where the user can fill in appropriate fields of the form. A given end user can access the database via one of the applications or can use an interface provided as an integral part of the database system software (such interfaces are also supported by means of applications, of course, but those applications are built-in, not user-written, e.g., query language processor). Naive users can read reports generated from the database.
2. **Application programmers :** Application programmers responsible for writing application programs that use the database. Application programmers are developers or computer professionals who write application programs. Application programmers develop user interfaces using any preferred language.

   Rapid Application Development (RAD) tools are available nowadays that enable an application programmer to construct application without writing code.

   Some programming languages combine control structures with database language statements. Such languages, sometimes called fourth-generation languages.
3. **Sophisticated users :** Sophisticated users interact with application without writing programs by using a database query language. This query will be solved by query processor.

   Online Analytical Processing (OLAP) tools is used to view summaries of data in different ways which helps analysts (e.g. sales of region, city etc.) with OLAP analysts can use data mining tools, which help them find certain kinds of patterns in data.
4. **Specialized users :** Creates the actual database and implements technical controls needed to enforce various policy decisions. Specialized users are sophisticated users who develop database applications. The DBA is also responsible for ensuring that the system operates with adequate performance and for providing a variety of other related technical services.

Q. 5 Explain different data models with its advantages and disadvantages. May 2014, May 2016

Ans. : Data Models

The data model will give the idea how final system or software will look like after development is completed. This concept is exactly like real world modeling in which before constructing any project (Bridges, Buildings, Towers etc.) engineers create a model for it, this model gives you the idea about how your project will look like after construction. A data model is an overview of a software system which describes how data can be represented and accessed from software system after its complete implementation.

Data models define data elements and relationships among various data elements for a specified system.

1. **Object Based Logical Models :** The data is stored in the form of objects, which are structures called classes that display the data within it. The fields are instances of these classes called as objects. This model is used in file management systems.

   The DBMS (Database Management System) developed with help of such model is called as OODBMS (Object Oriented Database Management System). Object oriented databases evolved to handle more complex applications such as databases for scientific experiments, geographic information system, engineering design and manufacturing. This model represents DB in terms of objects, their attributes and their behaviours.

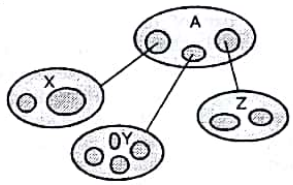


Fig. 1.1 : Object model

Advantages

(i) OO (Object Oriented) features provide a clear modular structure which is good for defining abstract data types where internal implementation details are hidden.

(ii) This model is easy to maintain and modify existing code as we can create new model with small change in existing.

Disadvantages

(i) This model is often provided through object oriented languages such as C++ and Java.

(ii) Practically very complex and inapplicable many a times.

2. **Record Based Logical Models**

The relational model first proposed by **E. F. Codd** hence he is known as father of Relational Model. A relational database is a collection of 2-dimensional tables which consists of rows and columns.

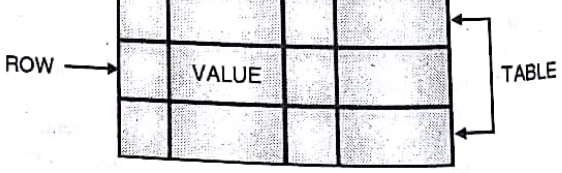


Fig. 1.2 : Relational model

Example : Most of the popular commercial DBMS products like Oracle, Sybase, MySQL etc. are based on relational model.

Advantages

(i) **Relational algebra :** A relational database supports relational algebra and various operations of the set theory (like union, intersection etc.)

(ii) **Dynamic views :** In a RDBMS, a view is not a part of the physical schema, it is always dynamic. Hence changing the data in a table also changes the data present in view.

(iii) **SQL (Structured Query Language) :** For data access in RDBMS we have English like query language called as structured Query language (SQL) which can be used for accessing data from RDBMS. Most of the database vendors support the SQL standard.

(iv) **Excellent data security :** Relational databases support the concept of user rights (every user is assigned with some database permission called as user rights), thus meeting the security needs of databases.

3. Hierarchical Model / Tree Model / XML Based Data Model

This was developed by joined efforts of IBM and North American Rockwell known as Information management system. It was the first DBMS model.

The data is sorted hierarchically, either in top down or bottom up approach of designing. This model uses pointers to navigate between stored data. This model represents data as a hierarchical tree.

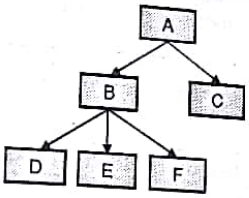


Fig. 1.3 : Hierarchical model

Example : One of the popular DBMS based on hierarchical model is Information Management System (IMS) from IBM.

Advantages

(i) **Conceptual simplicity :** Relationship between various levels is logically very simple. Hence database structure becomes easier to view.

(ii) **Database security :** Security is given by DBMS system itself it does not depends on whether programmer has given security or not.

(iii) **Simple creation, updation and access :** Hierarchical model is simple to construct with help of pointers or similar concepts and very simple to understand also adding and deleting records is easy in tree structure using pointers. This file system is faster and easy data retrieval through higher level records in tree structure.

(iv) **Database integrity :** There is always Parent Child Association between different levels of records in files. Hence child record is attached with the parent record which maintains the integrity.

Disadvantages

(i) **Complex implementation :** Only data independence is not enough for designer and programmers to build database system they need to have knowledge of physical data storage which may be complex.

(ii) **Difficult to manage :** Any change in a location of data needs change in all application programs that accesses changed data. Data access is restricted by Pointer path.

(iii) **Lack of structural independence :** Change in database structure does not affects data access is called as structural independence. Advantage of data independence may be restricted by structural independence.

(iv) Complex application programming

(v) Programmers must know how physical data is stored in order to access data. Even programmer knows path of data storage.

4. Network Model

Like the hierarchical model, this model also uses pointers toward data but there is no need of parent to child association so it does not necessarily use a downward tree structure. This model used in network databases.

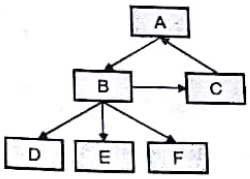


Fig. 1.4 : Network model

Example : IDS (Integrated Data Store) is one of the DBMS product based on network models. This was developed by joined efforts of IBM and North American Rockwell known as Information Management System.

Advantages

(i) Simple design : The network model is simple and easy to design and understand.

(ii) Ability to handle many types of relationship
1. The network model can handle the one-to-many or many-to-many or other relationships.
2. Hence network model manages multi user environment.

(iii) Ease of data access
1. In a network model an application can access a root (parent) record and all the member records within a set (child).
2. Provide very efficient and high speed retrieval.

Disadvantages

(i) System complexity
(a) In a network model, data are accessed one record at a time.
(b) This can increase the complexity of system for accessing multiple records at a time.

(ii) Lack of structural independence
Any changes made to the database structure (or data) require the application programs to be modified before it can access data.

Q. 6 What are the building blocks of data model.

Ans. : Basic Building Block of Data Model

1. **Introduction :** The basic building block for any of model is Entities, Attributes, relationships and constraints. This Model uses collection of tables to represent relationships amongst the data.

2. **Entity :** A fundamental component of a model. An entity is having its own independent existence in real world.
Example, A Student, Faculty, Subject having independent existence.
An entity may be an object with a physical existence or it may have logical existence.

Example, Entities like Department, Section, adult (age>18) may have physical existence or it may have only logical existence.

3. **Attributes :** Each entity has its own properties which describes that entity such properties are known as **attributes.** In relational model, the column in relation (Table) or field of data is also called as Attribute.

The single attribute will contains the similar type of data of all entities in relation.

Name
Mahesh
Suhas
Jay
Sachin

Example, The Name attribute in above student relation will contains the name of all student entities in student relation.

4. **Relationships :** A relationship is an association among several entities. E.g. Employee works for Department.

Degree : The degree of relationship type is number of participating entity types in a particular relation. Data model uses three types of relationships as below :

(a) **One is to one**
   1. One entity is associated with at most one other entity.
   2. **E.g.** One department can have only one manager.

(b) **One is to Many**
   1. One entity is associated with any number of entities in other entity.
   2. **E.g.** One teacher may teach to many students.

(c) **Many is to Many**
   1. One entity is associated with any number of entities in other entity.
   2. **E.g.** Books in library issued by students.

Q. 7 Compare various data models.

Ans. :

Sr. No.	Parameter	File system Model	Hierarchical Model	Network Model	Relational Model	ER Model	Object oriented Model
1.	Data Independence	No	Yes	Yes	Yes	Yes	Yes
2.	Structural Independence	No	No	No	Yes	Yes	Yes
3.	Storage Type	File	Segment	Record	Relation (Table)	Entity	Class
4.	Single row storage	Record	Segment occurrence	Current Record	Row(tuple)	Entity Occurrence	Object - instance of class
5.	Basic storage	Field	Segment field	Record field	Relation Attribute	Entity attribute	Object attribute
6.	Storage Identifier	Index	Sequence Field	Record key	Key	Entity key attribute	Object identifier
7.	Advantages	*Simple to implement, *Low cost implementation	*Promotes Data sharing, *conceptual simplicity, *Handle simple relationships(1:N) *Flexible data access	*conceptual simplicity, *Handle complex relationships (M:N)	*Tabular view, *Adhoc query capability, *improves management and implementation simplicity	*Very good *conceptual simplicity, *Effective *communication tool, *Integrated with dominant relational tools	*Semantic contents, *Promotes data integrity
8.	Disadvantages	*Limited implementations *Lack of standards	*Complex implementation, *Lack of standards *Limited implementations (No DML)	*Simplicity limits efficiency *Complex navigational system	*Hardware and software required	*Limited constraint representation *Limited relationship representation * No DML	* Slow development Of standards * complex navigation * Slow transactions if overload on system
9.	Examples	Operating system, Notepad, CSF(Comma Separated Files)	IMS (Information Management System)	IDS (Integrated Data Store)	Oracle, DB2, SQL SERVER etc.	ER diagrams	Database using C, java etc.

Q. 8 Write a note on : Database Schema and Instances.

Ans. :

1. **Database Schema :** Database schema is a structure denotes the logical view of complete database. Schema consists of entities and relationship among these entities. It is similar to programming data types and variable to store data. The database schema is the information about database or table structure.

Student table	Sid	Name	Class	Subject

Fig. 1.5 : Student Database Schema

Types of Database Schemas, Physical Database Schema. This schema represents physical structure of data or actual storage of data. It is similar to actual variable in programming. It defines how the data will be stored in memory.

Example,

The name is stored as a character data in storage. Logical Database Schema. This schema shows the logical structure need to be applied on the data stored. Similar to as data type of variable. It can be defined as tables, views, and integrity constraints.

Example,

The database consists of information about a set of students and departments in a college and the relationship between them

2. **Database Instance**

The data content of the database at a particular point in time is also referred as a database instance. It is similar to the value of a variable,

Student table	A001	Dinesh	FYJC	DBMS

Fig. 1.6 : Student Database Instance

It is always possible that this data will change with time. So it is very dynamic in nature.

Q. 9 State and explain various levels of database abstraction.

Ans. :

Three-Levels Schema Architecture

The goal of the three-schema architecture is to separate the front end (user applications interface) and the back end (physical database). The three-schema architecture is a tool with which the user can visualize the schema levels in a DBMS. Many DBMS systems do not separate the three levels completely, but support the three schema architecture to some extent. A description of data in terms of a data model is called a schema. The description of a database is called **database schema**, which is specified during database design and it does not expected to change frequently.

Database architecture

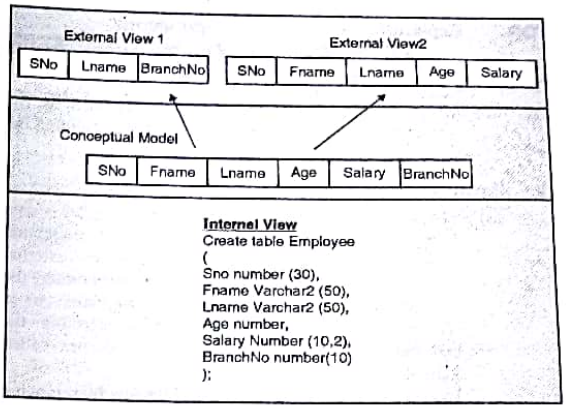


Fig. 1.7 : Database schema levels

(I) **Internal level (physical level) :** The internal level is very close to physical storage of data. This level describes the physical storage structure of the data in database. The internal (or physical) database is stored on secondary storage devices, mainly the magnetic disk. Describes the complete details of data storage and various available access methods for the database. At its ground level, it is stored in the form of bits with the physical addresses on the secondary storage device. At its highest level, it can be viewed in the form of files and simple data structures.

Internal view/ schema

The internal view defines the various stored data types and specified what indexes exist, how that stored fields are represented and so on. The internal schema uses a physical data model.

(II) **Conceptual level :** This level describes the structure of the whole database for a group of users. The conceptual model is also called as the data model or we can say data model is used to describe the conceptual schema when a database system is implemented. The conceptual schema hides the internal details of physical storage and targets on describing entities, data types, relationships and constraints. The conceptual schema contains all the information to build relevant external records. As the conceptual model is derived from the physical model. Conceptual view / schema. The conceptual view is a representation of the entire content of the database. The conceptual view includes definitions of each of the various conceptual data types.

(III) **External level (view level) :** The external level is the one closest to the user, i.e., it is the related with the way data is viewed by individual end users. The external level includes a number of user views or external schemas. Each external schema describes the segment of the database that is required for a particular user group and hides the rest of the database from that user group. External views are the proper interface between the user and the database, as an individual user can hardly be expected to be interested in the entire database. The external model is derived from the conceptual model.

Q. 10 Define data Independence and explain types of data Independence.

OR Explain the term : Data Independence.

Dec. 2013, Dec. 2014, May 2016, Dec. 2016

Ans. : Data Independence

1. **Definition :** Data Independence can be defined as the capacity to change one level of schema without changing the schema at the next higher level.

2. **Types**

(a) **Logical data independence :** Logical data independence is a capacity to change the conceptual schema without having any changes to external schemas. (or application programs). Separating the external views from the conceptual view enables us to change the conceptual view without affecting the external views. This separation is sometimes called logical data independence.

Example : Change the conceptual schema by removing a data item. In this case the external schemas that refer only to the remaining data should not be affected.

(b) **Physical data independence :** Physical data independence is a capacity to change the internal schema without having any changes to conceptual schema. The separation of the conceptual view from the internal view enables us to provide a logical description of the database without the need to specify physical structures. This is often called physical data independence.

Example : By creating additional access paths to improve the performance of retrieval. If the same data as before remains in the database, should not have to change the conceptual schema.

Q. 11 Write short notes on : Database Administrator.

Ans. : Database Administrator (DBA)

The database administrator is responsible for the overall planning of the company's data resources, for the design of data, and for the day-to-day operational aspects of data management. A database administrator is a person responsible for the installation, configuration, up gradation, maintenance and monitoring databases in an organizati on. The overall planning of corporate data is the strategic aspect of the database administration function and involves company-wide planning of existing data and assessment of organization-wise data standards.

Q. 12 Discuss the role of Database Administrator.

May 2016

Ans. : Roles of DBA

1. The DBA needs to performs many roles to keep the database up and running,
2. System Administrator / Designer
3. The database administrator need to manage DBMS software and server.
4. He is also responsible for deciding on the storage and access methods.
5. The DBA performs all data field updates or adding new fields into database.
6. Database Developer / Programmer
7. The DBA writes the programmes to design database and to design the means of reorganizing databases periodically.
8. The DBA also determine and implement database searching strategies.
9. System Analyst
10. The DBA needs to analyses the system performance and fine tune the DBMS activities.
11. DBA needs to take care of system crashes by planning proper recovery procedures.
12. He will also specify techniques for monitoring database performance

Q. 13 Write short notes on : Responsibilities of Database Administrator.

May 2012

Ans. : Responsibilities of DBA

The various responsibilities of DBA are as follows,

(i) **Designing overall Database schema :** The DBA is responsible for designing overall database schema (tables and fields). Also responsible for deciding on the data storage and access methods.

(ii) **Selecting and installing database software and hardware :** The DBA selects the suitable DBMS software like Oracle, SQL Server or MySQL. The Designing the means of reorganizing databases periodically.

(iii) **Designing Authorization/Access Control :** The DBA will decide the user access levels and security checks for access and data manipulations.

(iv) **Designing Recovery Procedures :** In order to take care of system crashes DBA needs to design the system recovery procedures and also specifying techniques for monitoring database performance.

(v) **Operations Management :** The operations management of database administration deals with data problems arising on a day-to-day basis. Specifically, the responsibilities include

(a) Investigation of errors found in the data.
(b) Supervision of restart and recovery procedures in the event of a failure.
(c) Supervision of reorganization of databases.
(d) Initiation and control of all periodic dumps of data.

Q. 14 Draw and explain database system structure.

Dec. 2013, May 2015, May 2017

Ans. : DBMS Architecture

A database system can be separated into two different modules that deal with all operations of the overall system.

Components of a database system,

1. Query Processor Components
2. Storage Manager/ Storage Management
3. Transaction Management

The storage manager is important because databases typically require a huge amount of storage space.

Query Processor Components

1. **Introduction :** The query processor will accept query from user and solves it by accessing the database.
2. **Parts of query processor**
   (i) DDL interpreter
   (ii) DML compiler
   (iii) Query evaluation engine

(i) **DDL interpreter :** This will interprets DDL statements and fetch the definitions in the data dictionary.

(ii) **DML compiler :** This will translates DML statements in a query language into low level instructions that the query evaluation engine understands. A query can usually be

translated into any of a number of alternative evaluation plans for same query result DML compiler will select best plan for **query optimization.**

(iii) **Query evaluation engine :** This engine will execute low-level instructions generated by the DML compiler on DBMS.

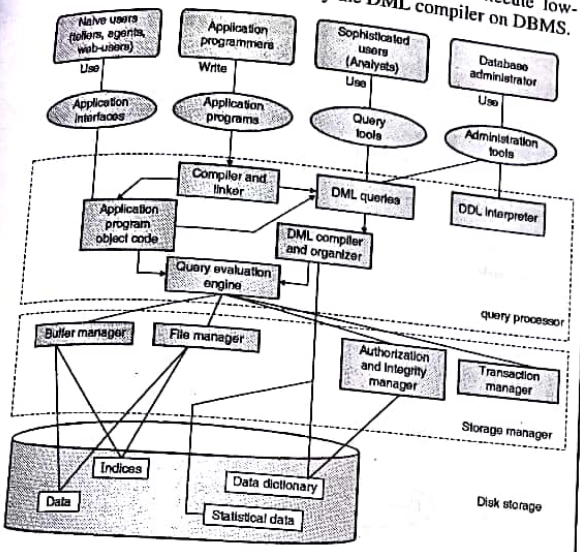


Fig. 1.8 : Components of DBMS

Storage Manager / Storage Management

A **storage manager** is a program module which acts like interface between the data stored in the database and the application programs and queries submitted to the system. The data is stored on the disk using the file system. The storage manager is programme which is responsible for the interaction with the file manager. The storage manager translates the various databases language statements into low level file system commands. Thus, the storage manager is responsible for storing, retrieving and updating data in the database.

The storage manager components include :

(i) **Authorization and integrity manager :** Checks for integrity constraints and authority of users to access data.

(ii) **Transaction manager**, which ensures that the database remains in a consistent (correct) state although there is system failures.

(iii) **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

(iv) **Buffer manager**, which is responsible for retrieving data from disk storage into main memory. The buffer manager is an important part of the database system, as it enables the database to handle data sizes that are much larger than the size of main memory. Data structures implemented by storage manager,

(a) **Data files :** Stored in the database itself.

(b) **Data dictionary :** Stores metadata about the structure of the database.

(c) **Indices :** Provide fast access to data items.

Transaction Management

A transaction is a series of small database operations that together form a single large operation. A transaction is started by issuing a BEGIN TRANSACTION command. Once this command is executed the DBMS starts monitoring the transaction. All operations executed after a BEGIN TRANSACTION command are treated as a single large operation. Application programs use transactions to execute sequences of operations when it is important that all the operations are successfully completed.

Transaction management component will ensure the atomicity and durability properties.

Chapter 2 : Entity Relationship Data Model

Q. 1 Define entity and entity set.

Ans. : Entity

Entity is anything in real world which may have physical or logical existence. An **entity** is anything in real world with its physical existence. **Example**, Student, faculty, subject having independent physical existence. An entity may be an object with a physical existence or it may have logical existence. **Example,** Department, Section, subject may have logical existence. Each entity has its own properties which describes that entity such properties are known as **attributes.**

Entity Type

Entity set is collection of entities with same attributes. As in Student table, each row is an entity and have same attributes. In other words we can say a student table is an entity type

The types of entities are,

(a) Strong entity type (b) Weak entity type

Entity Set

Entity set is collection of entities of same type. Example, Student entity set contains all students in college database.

Q. 2 What is strong entity ? Explain with example.

May 2012, May 2014

Ans. : Strong entity type

Entity type which has its own key attributes by which we can identify specific entity uniquely is called as strong entity type.

Example,

In case of Employee entity any specific employee can be identified by his Employee_id which is primary key of employee entity. In case of student in class each student identified by unique roll number which is his primary key.

Strong entity type is represented by single rectangle.

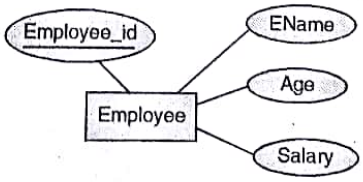


Fig. 2.1 : Employee entity

Q. 3 What is weak entity ? Explain with example. May 2012

Ans. : Weak entity type : Entity type which cannot form distinct key from their attributes and takes help from corresponding strong entity is called as weak entity type. These types of entities are dependent on strong entity for primary key. For some weak entities assign virtual primary key. Such virtual primary key of weak entity is called as **'discriminator'**. Weak entity type is represented by double rectangle.

Example : In case of "Dependent" entity depend on employee entity for primary key.

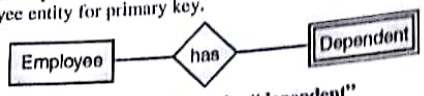


Fig. 2.2 : Weak entity "dependent"

Q. 4 Explain different types of attributes in ER Model. Dec. 2014, May 2017

Ans. : Attribute

Each entity has its own properties which describes that entity such properties are known as attributes. The attribute value that describes each entity becomes a major part of data stored in database. Employee entity may be described by attributes name, age, phone etc.

Type	Notation
Attribute (Simple/Single valued/Stored)	

A particular entity will have some value for each of its attributes.

The various types of attribute are used in ER diagrams,

(a) Composite Attributes : The attributes which can be divided in multiple subparts.

Type	Notation
Composite attribute	

The divisible attributes are composite attributes.

Example, The Name attribute of Student table can be divided into First_Name and Last_Name.

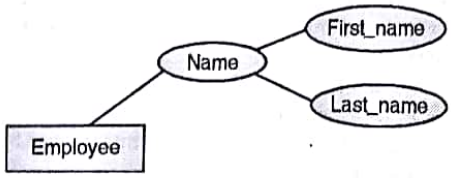


Fig. 2.3 : Composite attributes

(b) Multivalued attributes : The attribute having more then one value for a same entity is called as multi-valued attribute.

Type	Notation
Multivalued Attribute	

Example : A single student can have multiple mobile numbers.

Fig. 2.4 : Multi valued attributes

(c) Derived Attributes : The value of some attribute can be derived from the value of related stored attribute such attributes are known as derived attributes.

Type	Notation
Derived attribute	

Example, Employee tenure can be calculated from stored attribute 'Date_of_joining' of employee by subtracting it from today's date.

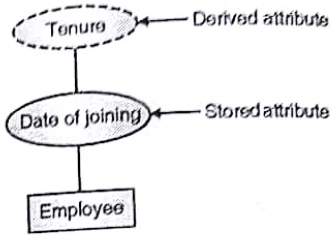


Fig. 2.5 : Derived attributes

(d) Null attribute : This attribute can take NULL value when entity does not have value for it. This is a special attribute the value of which is unknown, unassigned, not applicable or missing.

Example : The 'Net_Banking_Active_Bin' attribute gives weather particular customer having net bankig facility activated or not activated.

For bank which does not offer facility of net banking in customer table 'Net_Banking_Active_Bin' attribute is always null till Net banking facility is not activated as this attribute indicates Bank offers net banking facility or does not offers.

These attribute can be used in future use or for unknown, unsigned, missing values of attribute.

(e) Key attributes : This is an attribute of an entity which must have a unique value by which any row can be identified is called as key attribute of entity.

Example : Emp_Id for employee.

Fig. 2.6 : Key attributes

Type	Notation
Key attribute	Emp_id, Employee

The column value that uniquely identifies a single record in a table called as key of table. An attribute or set of attributes whose values uniquely identify each entity in an entity set is

called a key for that entity set. ID is a key of student table. It is possible to have only one student with a one ID (Say only one student 'Mahesh' with ID = 1)

Q. 5 What is relationship set ? Give various constraints of relationship.

May 2015, May 2016, Dec. 2016

Ans. : Relationship Set

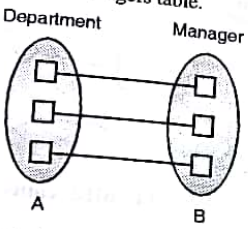
Collection of all relationship of same type is relationship set. The many employees are working for different departments so it is relationship set of Works_For relationship.

Types of mapping constraints

(i) One is to one : In this type of constraint one tuple in entity is related with only one tuple in other entity. That is one row in table is related with only one row in other table. A associated with at most one entity in B, B associated with at most one entity in A.

Example : one department can have only one manager.

Every row in Department table can be having relationship with only one row in Managers table.



(a) One to one mapping

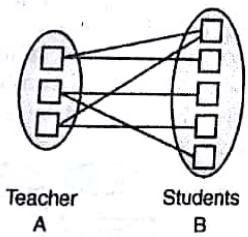
(b) Representation in ER diagram

Fig. 2.7 : One to one mapping

(ii) One to many : In this type of constraint one tuple in entity can be related with many tuples in other entity. A associated with any number of entities in B. B associated with at most one entity in A.

Example : One teacher may teach to many students.

Every row in Teacher table can have relationship with many rows in Student table.



(a) One to many mapping

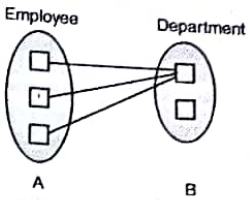
(b) Representation in ER diagram

Fig. 2.8 : One to many mapping

(iii) Many to one : In this type of constraint many tuple in entity can be related with only one tuple in other entity. A associated with at most one entity in B. B associated with any number of entities entity in A.

Example : Number of employee works for department.

Multiple rows in Employees table can be related with only one row in Department table.

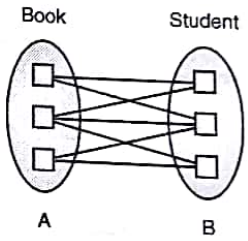


(a) Many to one mapping

(b) Representation in ER diagram

Fig. 2.9

(iv) Many to many : In this type of constraint many tuple in entity can be related with multiple tuples in other entity. A associated with any number of entities in entity B. B associated with any number of entities entity in A.



(a) Many to many mapping

(b) Representation in ER diagram

Fig. 2.10

Example : Books in library issued by students.

Multiple rows in Book table can be related with many rows in Student table.

Q. 6 Explain the terms total participation and Partial participation with example.

May 2015, May 2016, Dec. 2016

Ans. :

(i) Total participation : In case of total participation every object in an entity must participate in a relationship. The total participation is indicated by a dark line or double line between entity and relationship.

Example : Every department must have a manager.

Fig. 2.11 : Total participation

(ii) **Partial participation :** In case of partial participation more than one object in an entity may participate in a relationship. The total participation is indicated by a single line between entity and relationship.

Example: Employees works for department.

Fig. 2.12 : Partial participation

Q. 7 Explain Specialization with the help of an example.

Dec. 2013, May 2014, Dec. 2014, May 2016, Dec. 2016

Ans. :

Specialization

Top down approach of superclass / subclass relationship. Specialization is a process of defining a set of subclass of entity type, this entity type is called super **class of specialization.**

The set of subclasses that forms a specialization is defined on the basis of some distinguishing characteristic of entity in super class.

Example

Set of subclass (Saving_Account, Current_Account) are Specialization of super class Account.

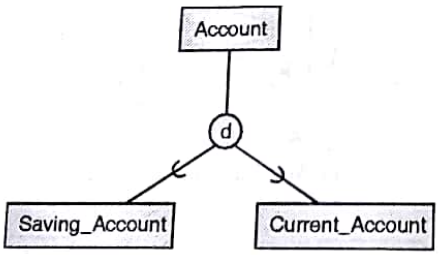


Fig. 2.13 : Specialization

Notation : The subclass defined in a specialization is attached by lines to a circle which is connected to super class. The subset symbol on each line connecting a subclass to circle indicates the direction of super class / subclass relationship.

Specific attribute

An attribute applied only to entities of particular subclass is called as specific attribute.

Summary of specialization
(i) Defines a set of sub class of an entity type.
(ii) Establish additional specific attributes with each subclass.
(iii) Establish additional specific relationship types between each subclass and other entity type or other subclass.

Q. 8 Explain Generalization with the help of an example.

May 2012, Dec. 2013, May 2014, May 2016, Dec. 2016

Ans. : Generalization

This is reverse process of specialization or this is bottom up approach of Superclass /subclass relationship.

Definition

Generalization is a process in which differentiate among several entity types identifying there common features and generalizing them to a single super class of which original entity type are special subclass.

Example

Car and Bike all having several common attribute they can generalize to the super class vehicle.

Notation

A diagrammatic notation to distinguish between generalization and specialization is used in some programming methodologies. Arrow pointing to generalized superclass represents generalization. Arrow pointing to generalized subclass represents specialization.

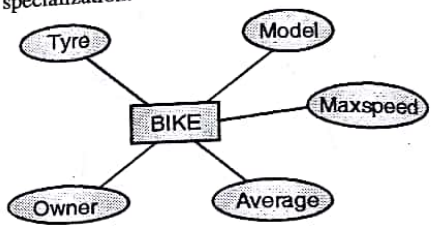


Fig. 2.14 : BIKE entity

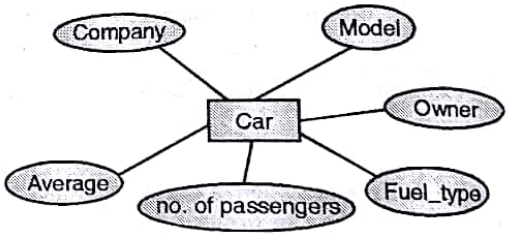


Fig. 2.15 : Car entity

Q. 9 Explain aggregation with the help of an example.

May 2012, May 2014

Ans. : Aggregation

Aggregation is meant to represent a relationship between a whole object and its component parts. It is used when we have to model a relationship involving entity sets and a relationship set. Aggregation allows us to treat a relationship set as an entity set for purpose of participation in (other) relationships.

Example : A Project is sponsored by a department. This is a simple relationship.

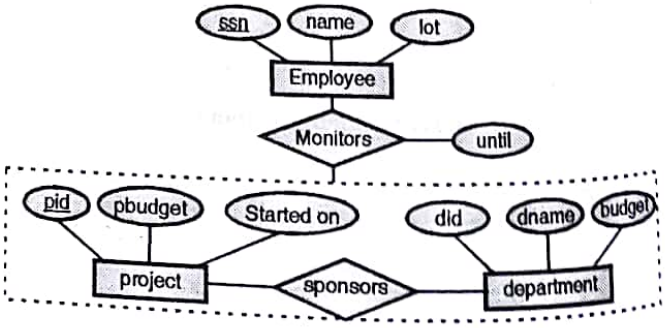


Fig. 2.16 : Aggregation

An Employee monitors this sponsorship (and not project or department). This is aggregation. Monitors are mapped to the table like any other relationship set.

Q. 10 A publication may be a book or an article. Articles are published in Journals. Publication has title and location. Book having their title and category. Article includes title, Topic and date. Publication is written by Authors stores Name, address and mobile number. Publication also belongs to particular subject which has their names.

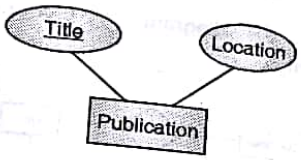
Ans. :

Step 1 : **Identify entities**

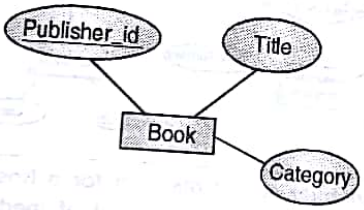
1. Publication
2. Book
3. Article
4. Journal
5. Subject
6. Author

Step 2 : **Identify attributes**

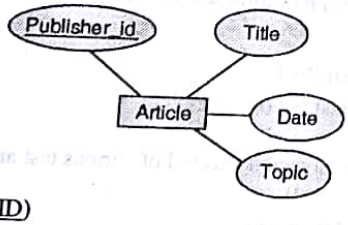
1. Publication (Title, Location)



2. Book (Publisher_id, Title, Category)

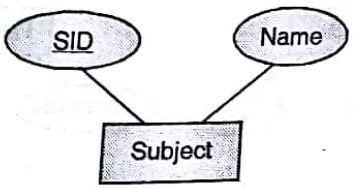


3. Article (Publisher_id, Title, Date, Topic)

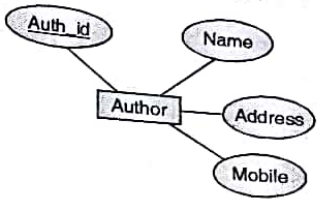


4. Journal (JID)

5. Subject (SID, Name)



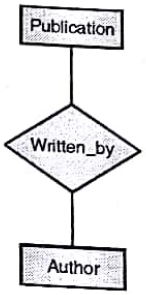
6. Author (Auth_id, Name, Address, Mobile)



Step 3 : **Identify relationships**

1. Articles are published in Journal.

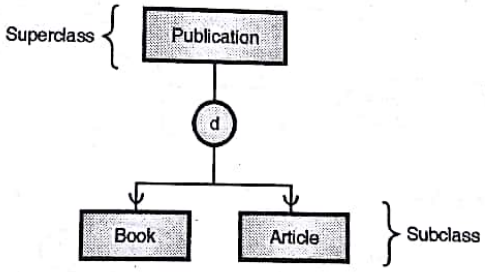
2. Publication is written by Author.



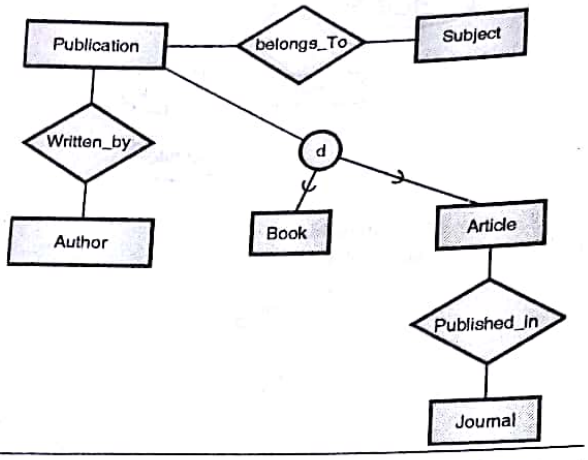
3. Publication belongs to a particular subject.

Step 4 : **Identify inheritance relations**

Publication can be
BOOK or ARTICLE.



Step 5 : Merging all above relations we will get final ER model



Q. 11 Construct on E-R diagram for a car-insurance company that has a set of customers each of whom owns one or more cars. Each car has associated with it zero to any number of recorded accidents. May 2014, May 2015

Ans. :

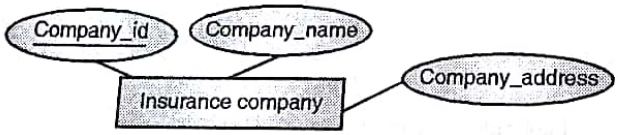
(1) Identify all entities

(a) Insurance company (b) Customer

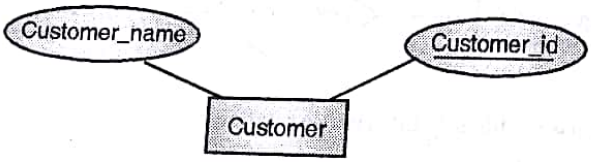
(c) Car (d) Accidents

(2) Identify all attributes

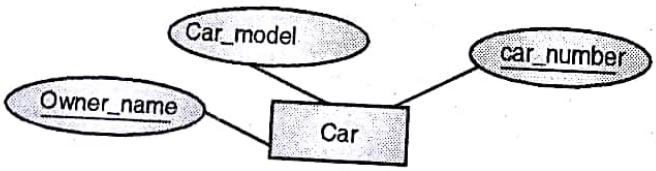
(a) Company entity



(b) Customer entity



(c) Car entity



(d) Accidents

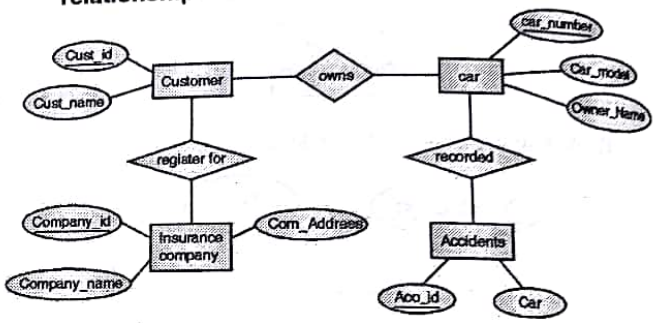
(3) Identify all relationship

a) Car insurance company has a set of customers

b) Customer owns one or more car

c) Each car associated with zero or any number of accidents

(4) Construct ER diagram by merging all above relationships



Q. 12 Construct an ER diagram for a hospital with a set of patients and the set of medical doctors associated with each patient a record of various text and examination conducted.

Ans. :

(1) Identify Entities

(i) Hospital (ii) Patient

(iii) Doctors

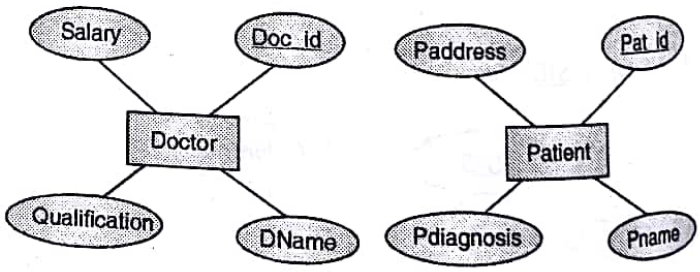
(iv) Medical-record (Record of various test and examination conducted)

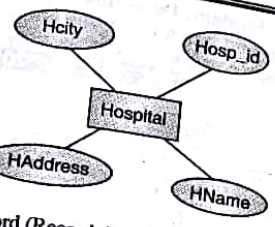
(2) Identify Attributes

(i) Hospital (Hosp_id, HName, HAddress, Hcity)

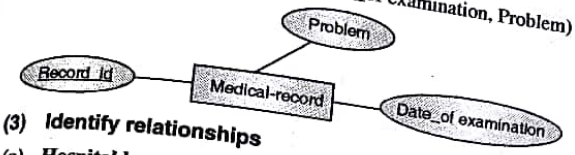
(ii) Patient (Pat_id, Pname, Pdiagnosis, Padress)

(iii) Doctor (Doc_id, DName, Qualification, salary)





(iv) Medical_Record (Record_id, Date_of examination, Problem)



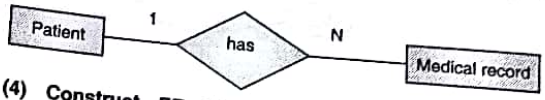
(3) Identify relationships

(a) Hospital has a set of patients

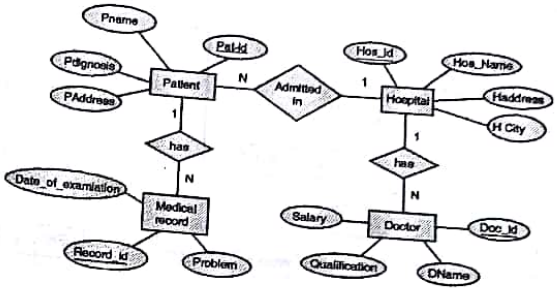
(b) Hospital has a set of doctors

(c) Doctors are associated with each patient

(d) Each patient has record of various test and examination conducted

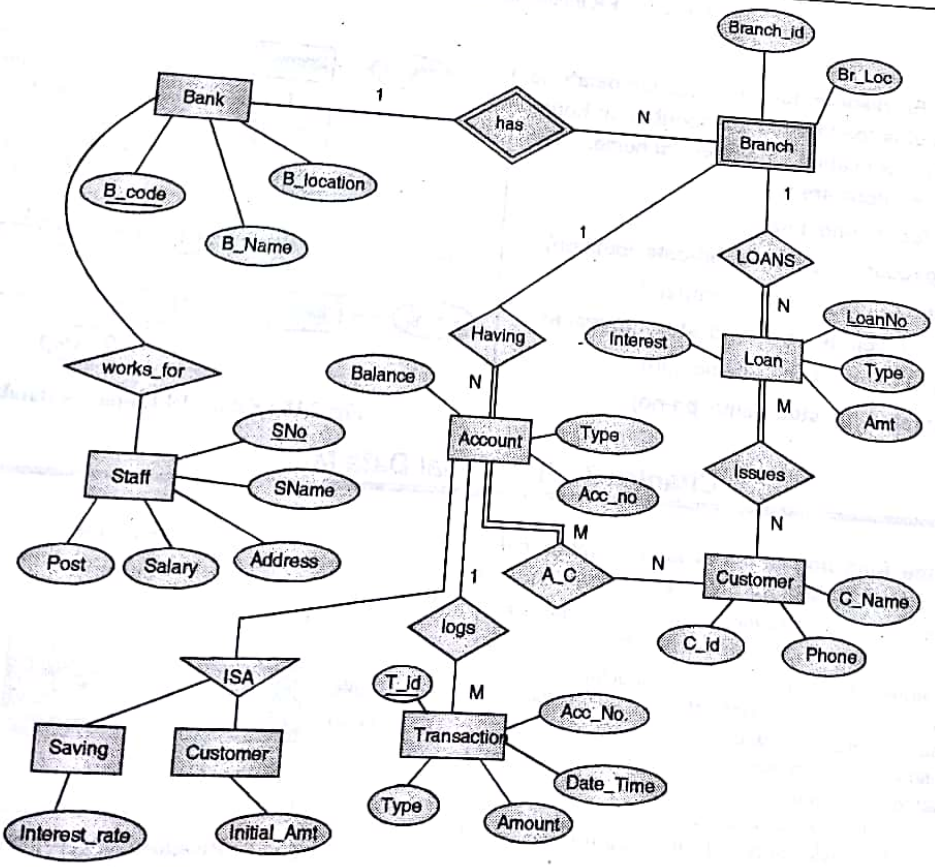


(4) Construct ER Model from merging all above relationship



Q. 13 Draw ER Diagram for banking enterprise.

Ans. :



Q. 14 **Draw ER Diagram for University database consisting four Entities Student, Department, Class and Faculty. Student has a unique id, the student can enroll for multiple classes and has a most one major. Faculty must belong to department and faculty can teach multiple classes. Each class is taught by only faculty. Every student will get grade for the class he/she has enrolled.** Dec. 2013, Dec. 2014, Dec. 2016

Ans. :

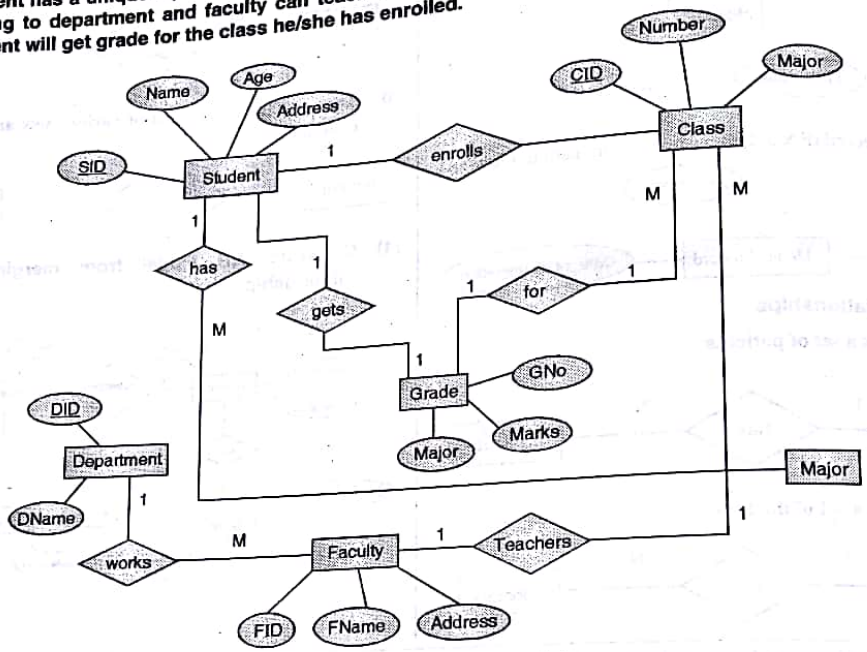


Fig. 2.17 : ER model for university database

Q. 15 **Draw an ER diagram for the education database that contains the information about an in house company education training scheme. The relevant relations are :**

Course (course-no, title)

Offering (course-no, off-no, off-date, location)

Teacher (course-no, off-no, emp-no)

Enrolment (course-no, off-no, stud-no, grade)

Employee (emp-no, emp-name, job)

Student (stud-no, stud-name, ph-no)

Ans. :

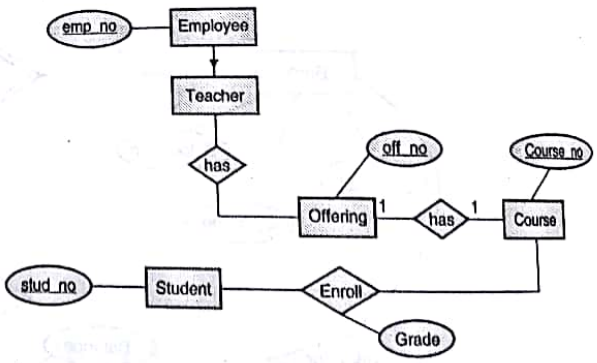


Fig. 2.18 : ER model Employee database

Chapter 3 : Relational Data Model

Q. 1 **Define the following terms related to relational model : Relation, Attribute, Tuple and Domain.**

Ans. : Tables are known as **relations**, columns are known as **attributes** and rows (or records) are known as **tuples**.

1. Relation (Table) : Relations are a logical structure which is a collection of tables consisting horizontal rows also called as tuples and vertical columns also called as Attributes. The table containing rows and columns represents entity in relational model, it is called as Relation. This concept doesn't represent how the data is stored in the physical memory of computer system. The relation can contain data about a single entity or relationship between two entities.

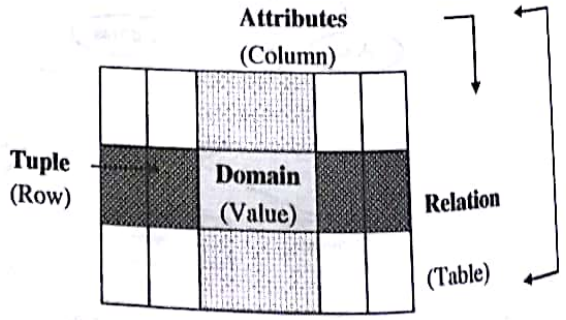


Fig. 3.1 : Relational Algebra Notations

Characteristics of Relation

1. A table composed of rows and columns.
2. Each table in a database has its unique *table name*.
3. Each table row (tuple) represents a single entity occurrence within the entity set.
4. All values in a same column must conform to the same format of data.
5. Each table must have a single attribute or set of attributes that uniquely identifies each row.

Example,

The student relation can be shown as given below,

Id	Name	Age	Class	Branch
105	Mahesh	25	BE	IT
106	Suhas	28	FE	CS
107	Jay	29	SE	CS
108	Sachin	30	TE	EXTC

2. **Attributes (Column) :** Relation has its own properties which describes that relation (table) such properties are known as attributes. In relational model, the column in relation (table) or field of data is also called as Attribute. Every table must have at least one column in it. The single attribute will contains the similar type of data of all entities in relation.

Name
Mahesh
Suhas
Jay
Sachin

It is not possible to have multiple columns with same column name in the same relation. But it is possible to have multiple columns with same column name in two different relations. The SQL standard does not specify any maximum number of columns in a table.

Example,

The Name attribute in above student relation will contains the name of all student entities in student relation.

3. **Tuple (Row / Records) :** A single row in relational table which contains all the information about a single entity is called as Tuple. The single row in relation (Table) is called as Tuple. Each horizontal row of the student table represents a Student tuple. A table can have any number of rows in it.

Stud_Id	Name	Age	Std	Div
105	Mahesh	25	BE	A

Example : The above tuple contains all data about the Id 105, student entity in student relation.

4. **Domain (Data Value) :** The intersection column and row in a relational table which represents data of entity is called as Domain. Every column in a table has a set of data values that are allowed for that column which is called as Domain. In a relational table a domain can have a single value or no (Null) value. The single domain will contains the specific data of single entities in relation.

Name
Mahesh

Example : The Name attribute of tuple id 105 will contains the name of all student with id 105 in student relation.

Q. 2 Write a note on Relational Database Schema.

Ans. :

Relational Database Schema

The term schema refers to the organization or structure of data in relational database. The relational schema describes structure of relation (i.e. table) and relational database schema explains the structure of relational database.

Relational Schema

Relation schema consists of a number of attributes associated with relation.

Example

Sample Relational Schema

EMPLOYEE	(Ssn, Ename, Bdate, Address, Dnumber)
DEPARTMENT	(Dnumber, Dname, Dmgr_ssn)
DEPT_LOCATIONS	(Dnumber, Dlocation)
PROJECT	(Pnumber, Pname, Plocation, Dnum)
WORKS_FOR	(Ssn, Pnumber, Hours)

Relational Database Schema

Relational database schema consists of a number of relation schemas associated with that database.

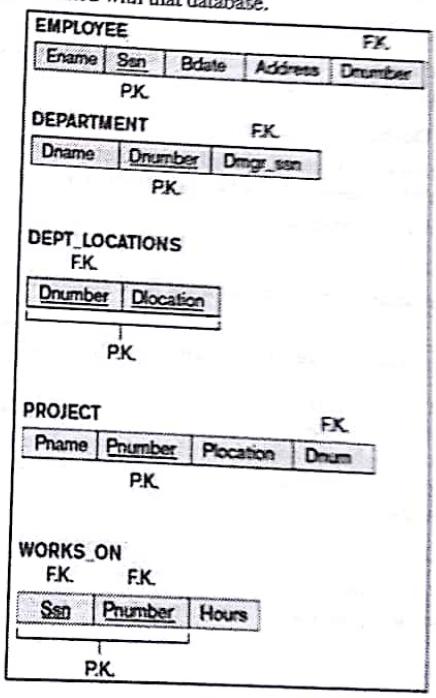


Fig. 3.2 : Sample Relational Database Schema

The attributes are grouped to form a relation schema by mapping a conceptual data model i.e. ER or EER data model to relational schema. The relational mapping will identify entity types and relationship types and their respective attributes. The relation schema consists of a number of attributes in relation while the relational database schema consists of a number of relation schemas in the corresponding database.

Relational Schema (Table Structure)	Relational Database Schema (Database Structure)
$R_1 (A_1, A_2, ..., A_n)$ $R_2 (B_1, B_2, ..., B_n)$ ... $R_m (C_1, C_2, ..., C_n)$	$R_1, R_2, ..., R_n$

Where,

$R_1, R_2, ... R_n$ = Relation or Tables

A, B, C = Attributes or Columns

Q. 3 Explain different integrity constraints. Dec. 2016

Ans. : Integrity Constraints

Constraints makes sure that only authorised user will make modifications to database and changes should not lead to loss of data consistency and correctness. These concepts makes sure correctness and completeness of data stored in the database. This objective can never be guaranteed, one cannot ensure that every entry made in database is accurate.

Types of Relational Constraints

1. Domain Relational Constraints
2. Entity Relational Constraints
3. Referential Relational Constraints

1. Domain Relational Constraint

Domain constraints allow us to test whether the values inserted into the database are correct or not. The CREATE TABLE Command may also include domain constraints which can check integrity of database. These domain constraints are the most basic form of integrity constraint.

Types of Domain Constraints

1. Nullness Constraint / Required Data Constraint
2. CHECK Constraint / User Defined Constraint
3. DEFAULT Constraint

Example 1

In the student table, student must have an associated student name.

```
Student_Name varchar(100) NOT NULL
```

Therefore now, the Student_Name column in the STUDENT table is a required data column. It is not possible to insert Null value in Student_Name column of Student table. The DBMS can prevent user from inserting NULL values in any column with help of such constraints.

Example 2

Table with student entity having gender which can be M or F.

Hence, attribute gender can take only two values either 'M' or 'F'.

```
Student_Gender varchar(1) CHECK (gender IN ('M','F'))
```

Example 3

Table with customer entity having name and gender.

If name is not added for customer that will be taken as 'Unknown' if specify DEFAULT value of NAME column to 'UNKNOWN'

```
Student_Name varchar(50) DEFAULT 'UNKNOWN'
```

2. Entity Integrity Constraints

Entity constraints allow us to test whether the tuple (entity) inserted into the database are correct or not.

The create table Command may also include entity constraints which can primary key of table.

Types of Entity Constraints

1. Unique Constraint 2. Primary Key Constraint

Example

```
EMAIL varchar(30) UNIQUE
```

3. Referential Integrity

A value appearing in a one relation (table) for a given set of attributes also appears for another set of attributes in another relation (table). This is called referential integrity. The referential integrity constraint is specified between two tables to maintain the consistency among tuples in the two tables. The tuple in one relation refers only to an existing tuple in another relation.

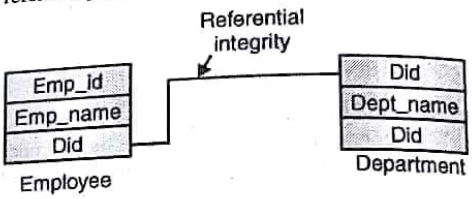


Fig. 3.3 : Referential integrity

Employee Table		
Emp_Id	**Emp_name**	**Did**
1	Sachin	20
2	Suhas	10
3	Jay	20
4	Om	10

Department Table	
Did	**Dept_name**
10	HR
20	TIS
30	L&D

In the above example Employee table has Did as foreign key reference to Did column in Department table this is called as referential integrity. Here, forcing the database to check the value of Did column from the department table while inserting any value in Employee table. This helps to maintain data consistency.

Q. 4 Explain the term : Primary Key. May 2016

Ans. :

Primary Key Constraint

A table in a relational database has one column or combination of some columns whose values uniquely identifies a single row in the table. This column or combination of columns is called the **primary key** of the table. Primary key attribute is same as unique key constraint with NOT NULL constraints (Unique constraint + Not Null constraint). The main difference in unique constraint and primary key constraint is that one null value is allowed in unique constraint which can be treated as unique value while nulls are not allowed in primary key constraint.

For example, each row of the STUDENT table has a unique set of values in its STUDENT_ID column, which uniquely identifies the student represented by that row.

Duplicate values are not allowed in primary key column, because they cause problems in distinguishing one entity from another (entity may be an employee). The DBMS can prevent user from inserting same data values in a column again and again.

```
STUDENT_ID char(10) PRIMARY KEY
```

Q. 5 Discuss what is meant by term : Referential integrity. Dec. 2015

Ans. : Referential Integrity / Foreign Key

A value appearing in a one relation (table) for a given set of attributes also appears for another set of attributes in another relation (table). This is called referential integrity. The referential integrity constraint is specified between two tables to maintain the consistency among tuples in the two tables. The tuple in one relation refers only to an existing tuple in another relation.

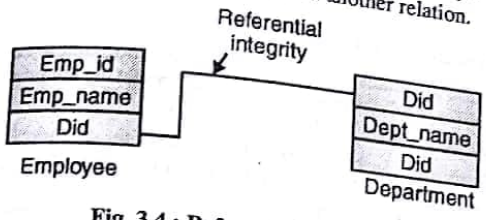


Fig. 3.4 : Referential integrity

Employee Table		
Emp_Id	**Emp_name**	**Did**
1	Sachin	20
2	Suhas	10
3	Jay	20
4	Om	10

Department Table	
Did	**Dept_name**
10	HR
20	TIS
30	L&D

In the above example Employee table has Did as foreign key reference to Did column in Department table this is called as referential integrity. Here, forcing the database to check the value of Did column from the department table while inserting any value in Employee table. This helps to maintain data consistency.

Foreign key violations in SQL

If any row in EMP table is added with 'Did' value which is not there in department table the insert statement will give foreign key violation error. In above tables we will refer Department as parent table (as it is containing Primary key) and Employee table as Child table (as it is containing Foreign Key). There are 4 problems causes the foreign key violations.

Adding new tuple to Child Table (Add Child). If we try to add an employee with Did 70 to employee table (Child Table), it will return foreign key violation error. As Did 70 is not there in Department table (Parent table).

```
INSERT INTO Employee
VALUES (11,'Devid', 70);
```

Output

ORA-02291 : integrity constraint (Employee.FK_Employee) violated - parent key not found

Adding new employee		
Emp_Id	**Emp_name**	**Did**
11	Devid	70

This functionality helps to maintain data consistency in database.

Updating tuple from Child Table

If we try to update an employee Emp_Id = 2 with Did as 70 to employee table (Child Table), it will return foreign key violation error. As Did 70 is not there in Department table (Parent table).

```
UPDATE Employee
SET Did = 70
WHERE Emp_Id=2;
```

Output

ORA-02291 : Integrity constraint (Employee.FK_Employee) violated - parent key not found. This functionality helps to maintain data consistency in database.

Deleting tuple from Parent Table

If we try to delete department of Did = 10 from Department table (Parent table), it will return foreign key violation error. As there are few employees working in department with Did =10.

```
DELETE Department
WHERE Did=10;
```

Output

ORA-02292: integrity constraint (Employee.FK_Employee) violated - child record found. This functionality will creates limitation for deletion of parent record if it has some associated child records.

Updating tuple from Child Table

If we try to update department of Did = 10 with Did = 70, it will return foreign key violation error. As there are few employees still working in department with Did =10.

```
UPDATE Department
SET Did = 70
WHERE Did = 10;
```

Output

ORA-02292 : integrity constraint (Employee.FK_Employee) violated - child record found. This functionality will creates limitation for updating parent record if it has some associated child records.

Q. 6 Explain concept of keys relational Table.

Ans. : Keys

The column value that uniquely identifies a single record in a table called as key of table. An attribute or set of attributes whose values uniquely identify each entity in an entity set is called a key for that entity set. ID is a key of student table. It is possible to have only one student with a one ID (Say only one student '**Mahesh**' with ID = 1)

Key type	Definition
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row.
Secondary key	An attribute (or combination of attributes) used strictly for data retrieval purposes.
Foreign key	An attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null.

Q. 7 Explain the steps of an algorithm for ER to relational mapping. Dec. 2013, May 2017

OR Explain the algorithm to map ER and EER model to relational model in detail.

Ans. :

Mapping Entities to Tables

(1) Regular entity types

Tables : Regular entity sets can be represented as table in relational model.

Columns : Attributes of entity set can be converted to the columns (attributes) of the tables in relational model.

Example : Regular entity employee mapped as employee table in object model like 'Stud_id', 'Stud_Addr' etc. are shown as table columns.

ER model

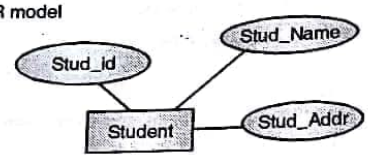


Fig. 3.5 : Regular entity

Stud_id	Stud_Name	Stud_Addr
1	Snehal	Mumbai
2	Pratiksha	Mumbai
3	Supriya	Mumbai
4	Tanmay	Goa

(2) Weak entity types

For each weak entity type with owner entity, create a table and include all simple attributes of weak entity type as columns of table, including foreign key attributes as the primary key attribute of the table that correspond to the owner entity type.

Example : Dependents (Weak entity) in Employee (Owner entity).

ER Model

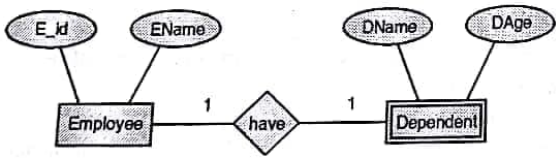


Fig. 3.6 : Weak entity

E_id	Ename	DName	DAge
1	Sachin	Jyoti	23
2	Suhas	Manju	22
3	Jayendra	Tanya	27

Q. 8 Explain how to convert attribute in ER to relational Table.

Ans. : Mapping Attributes to Columns of Table

(a) Simple attributes

Simple attribute can be directly converted to a column (Attribute) in relational model.

Example,

Employee 'Age' can be directly converted to column.

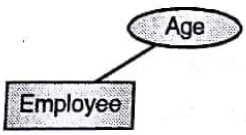


Fig. 3.7 : Simple attribute

Employee table

Eid	Age
1	23
2	24
3	43
4	28

(b) Composite attributes

These attributes need to be stored as set of simple component attributes (Columns) in relational model by avoiding actual attribute ('name' in below example).

Example : In below example composite attribute 'Name' is converted to three columns in object model by avoiding actual attribute 'Name'.

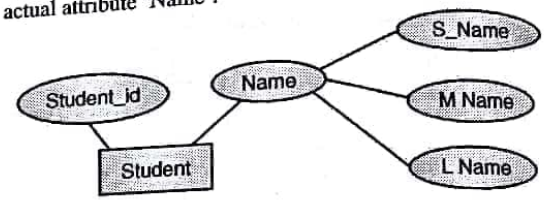


Fig. 3.8 : Composite attribute

Student table

Student_id	S_Name	MName	LName
1	Harshad	Rupali	Malar
2	Bipin	Anand	Shinde
3	Aanand	Ganesh	Panchal
4	Tushar	Bipin	Pimple

(c) Multi valued attributes

Multi valued attributes are mapped as a relation which includes combination of the primary key of table and multi valued attribute as a composite primary key as shown in Fig. 3.9.

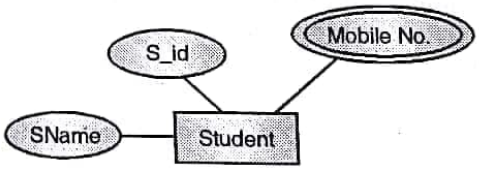


Fig. 3.9 : Multi valued attribute

Student table

Sid	SName
1	Jayendra
2	Suhas
3	Sachin

Mobile Table

Sid	Mobile No.
1	9891295492
1	9821959241
2	808091856
3	8095124890
3	9773112456

Mobile table

Sid	Mobile No.
1	9891295492
1	9821959241
2	8080918456
3	8095124890
3	9773112456

(d) Derived attributes :

There is no need to store such attribute in relational model. It will be calculated from stored attribute.

(e) Key attributes : Key attribute in ER Model can be directly converted to primary key attribute of relational model.

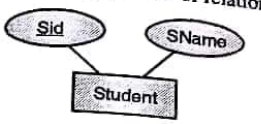


Fig. 3.10 : Key attributes

Student table

Sid	SName
1	Deepak
2	Vaibhav
3	Yogita
4	Bency

Q. 9 Explain how to convert various type of relations in ER to relational Table.

Ans. : Foreign key approach

If binary relationship type does not possess many attributes then map such relation using foreign key.

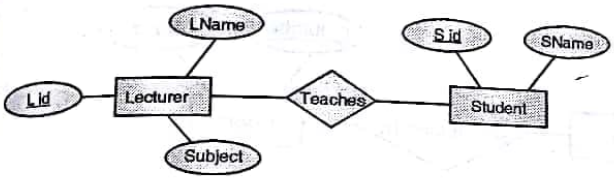


Fig. 3.11 : Foreign key approach

Lecturer table		
Lid	**LName**	**Subject**
1	Omprakash	IP
2	Yogesh	INS
3	Amit	PM

Student table		
Sid	**SName**	**Lid**
1	Bency	1
2	Deepak	1
3	Yogita	1
4	Snehal	2
5	Pratiksha	2

Lid is foreign key in student table while primary key in lecturer table.

Merged relationship approach

When participation is total it is possible to merge relation and involved entities as a single relation and then map it to a table.

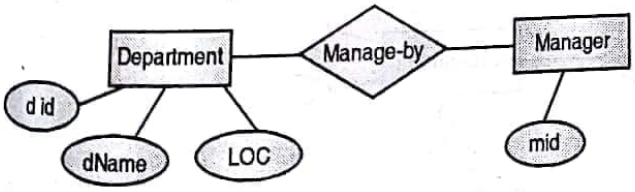


Fig. 3.12 : Merged relationship approach

Department table			
did	**Dname**	**Loc**	**mid**
10	IDF	Mahape	11
20	Mayban	Pune	22
30	Lax	Mumbai	33

Cross reference approach : A relationship type in EER is mapped to new table in relational model. Column of such table is all attributes of relation and primary key attributes of all tables linked to this relation.

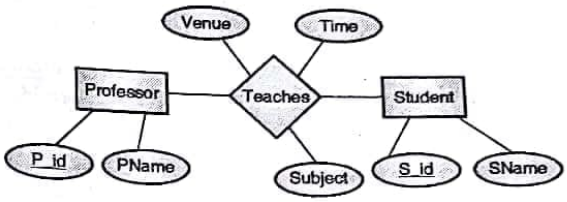


Fig. 3.13 : Cross reference approach

Professor table	
Pid	**PName**
1	Om
2	Nitin

Student table	
Sid	**SName**
1	Snehal
2	Tanmay

Teacher relation table				
Pid	**Sid**	**Venue**	**Time**	**Subject**
1	1	SFIT	1 pm	ADBMS
1	2	XIE	1 pm	DBMS

Q. 10 Explain how to convert inheritance in ER to relational Table.

Ans. : Mapping Inheritance constraints

Tables : Each super class and subclass entity sets represents table in relational model.

Columns : Attributes of entity set is converted to the columns of the tables in relational model.

Primary key : The primary key column of super class is also added to all subclasses and treated as a primary key column for all tables in relational model.

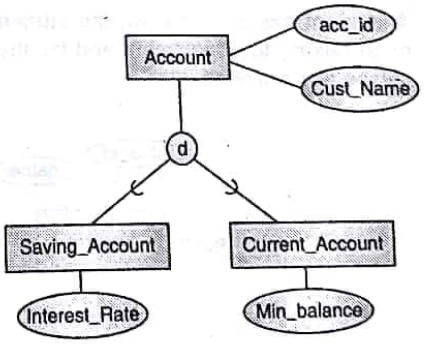


Fig. 3.14 : Inheritance relation

Account table	
acc_id	**Cust_Name**
1	Snehal
2	Tanmay
3	Nikhil

Saving_Account table	
acc_id	Interest_Rate
1	10
2	12
3	15

Current_Account	
acc_id	Min balance
1	1000
2	2000
3	500

Q. 11 **Draw an E-R diagram and reduce it to relational database model for a university database for scheduling of classrooms for final exams. This database could be modelled using entities as**
exam (course_name, section_number, room_number, time);
course. (name, department, C_number),
room (r_number, capacity, building).
Entity section is dependent on course.

Ans. :

***Step 1* : ER diagram**

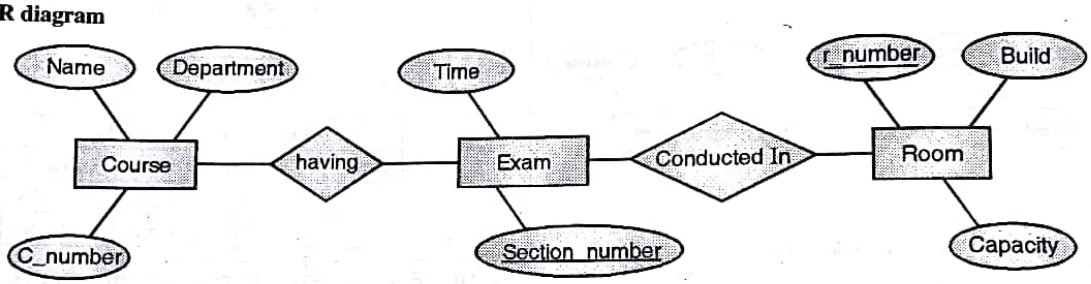


Fig. 3.15

Step 2 : Mapping to Tables

Course (C_number, name, department)
Exam (Section_number, time, C_number,)
Room (R_number, building, capacity, Section_number)

Q. 12 **Draw an E-R diagram for a university database consisting of 4 entities,**
(i) Student (ii) Department (iii) Class
(iv) Faculty and convert it to tables.
A student has a unique id, the student can enroll for multiple classes and has at most one major. Faculty must belong to department and faculty can take multiple classes-Every student will get a grade for the class he/she has enrolled.

Ans. :

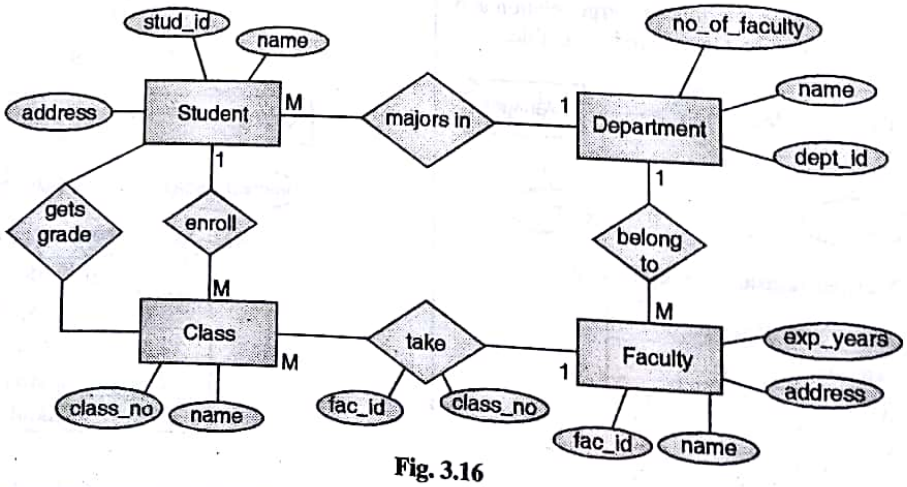


Fig. 3.16

Student	
Id	Primary Key
Name	
Address	
Dept_id	Foreign key references to dept_id column of Department table
Faculty	
Fac_id	Primary Key
Fac_name	
Exp_years	
Address	
Dept_id	Foreign key references to dept_id column of Department table
Class	
Class_ no	Primary Key
C_name	
Stud_class	
Class_ no	Foreign key references to dept_id column of Department table
Stud_ id	Foreign key references to dept_id column of Department table
take_class	
Fac_id	Foreign key references to fac_id column of Faculty table
Class_ no	Foreign key references to class_ no column of Class table
Department	
Dept_id	Primary Key
D_name	
No_of faculty	
Grade	
Stud_ id	Foreign key references to dept_id column of Department table
Class_ no	Foreign key references to dept_id column of Department table
Grade	

Q. 13 Construct on E-R diagram for a car-insurance company that has a set of customers each of whom owns one or more cars. Each car has associated with it zero to any number of recorded accidents. May 2014, May 2016

Ans. :

(1) Identify all entities

(a) Insurance company (b) Customer
(c) Car (d) Accidents

(2) Construct ER diagram by merging all above relationships

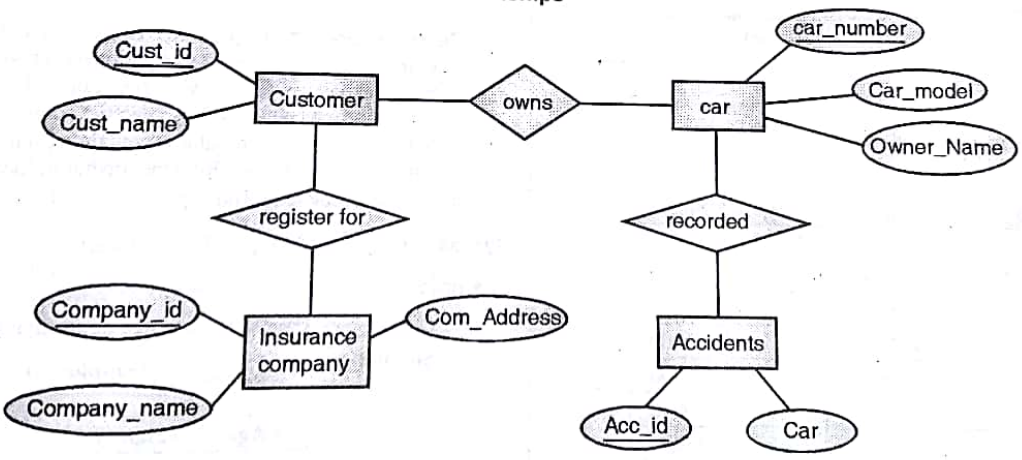


Fig. 3.17

(1) Mapping Entities

(a) Company (Company_id, Name, Address)
(b) Customer (Customer_id, Name, Address, phone)
(c) Car (Car_Number, Car_Model, Owner)
(d) Accidents (Accident_Id, Location, date, time)

(2) Mapping Relations

(a) Company (Company_id, Name, Address)
(b) Customer (Customer_id, Name, Address, phone, Insurance_Company)
Insurance_Company - refers to customers registered insurance company
(c) Car (Car_Number, Car_Model, Owner_Id)
Owner_Id - refers to customer id owns that car.
(d) Accidents (Accident_Id, Car_Number, Location, date, time)
Car_Number - refers to car involved in accident.

(3) Final Relational Schema

(a) Company (Company_id, Name, Address)
(b) Customer (Customer_id, Name, Address, phone, Insurance_Company)
(c) Car (Car_Number, Car_Model, Owner_Id)
(d) Accidents (Accident_Id, Car_Number, Location, date, time).

Chapter 4 : Relational Algebra

Q. 1 Explain any four relational algebra operations with proper examples.

May 2012, Dec. 2013, Dec. 2014, Dec. 2015, Dec. 2016, May 2017

Ans. : Relational Algebra Operation

1. Unary Relational Operations
   - Project operation (π)
   - Select operation (σ)
   - Rename operation (ρ)
2. SET Theory Operations
   - Union operation ($\cup$)
   - Difference operation (-)
   - Intersection operation ($\cap$)
3. Binary Operations
   - Join operations ($\bowtie$)
   - Cartesian product operations (X)
   - Division operation (%)

1. Selection Operation (σ) : This operator is used to select some rows from table which satisfy particular selection condition given in selection operation. Selection operator selects a set of tuples that satisfy a selection predicate or condition. Output of query is exactly same as input schema of table. This is unary relational operator having only one input table.

Syntax :

σ <attribute_name><comparison_operator><constant_value> (Input_Table_Name)

Where,

Attribute_name : Name of column in table

Comparison_operator : =, <, < =, >, > =, < >

Example :

Eid	Ename	Age	Salary
1	Suhas	24	50000
2	Jayendra	24	15000
3	Sachin	25	52000
4	Mahesh	23	41000
5	Satish	34	25000
6	Suma	54	50000
7	Raj	69	45000
8	Anu	74	50000

Query : Select all employees having age below 30 years.

Solution : $\sigma_{age < 30}$ **(Employee)**

Eid	Ename	Age	Salary
1	Suhas	24	50000
2	Jayendra	24	15000
3	Sachin	25	52000
4	Mahesh	23	41000

2. Projection Operation (π) : This operator is used for selecting some of many columns in table to display in result set. Projection operator can select a column or set of columns of table to be display in output of query. Select only few columns or all columns of a table as per requirements. This is unary relational operator having only one input table.

Syntax : $\pi_{< column\_list >}$ (Input_Table_Name)

Example :

Query : Find salary and age of all Employees.

Solution : $\pi_{age, salary}$ **(Employee)**

Age	Salary
24	50000
24	15000
25	52000
23	41000
34	25000
54	50000
69	45000
74	50000

Query : Find salary of all employees having age less than 25.

Solution : $\pi_{age, salary}$ **(Employee)**

Age	Salary
24	50000
24	15000
23	41000

3. Rename Operation (ρ) : Give alternative name to any column (attribute) or any table of query expressions using operator called as RENAME operator. This operator is specially introduced to select specific column from joined table (set of two or more tables) containing multiple columns of same column name. Rename operator, denoted by the lowercase Greek letter rho (ρ).

Syntax : $\rho_{< New\_Name >}$ (Input_Table_Name)

Example :

Query : Find salary and age of all Employees.

Solution : $\pi_{e.age, e.salary}$ **(ρ_e (Employee))**

e.Age	e.Salary
24	50000
24	15000
25	52000
23	41000
34	25000
54	50000
69	45000
74	50000

4. Union Operator : This operator finds out all combined rows in table 1 and table 2. Union effectively appends the result of first query to the result of second query. It does not eliminate all duplicate rows and they are printed in result expression.

Syntax : (Query Expression 1) $\cup$ (Query Expression 2)

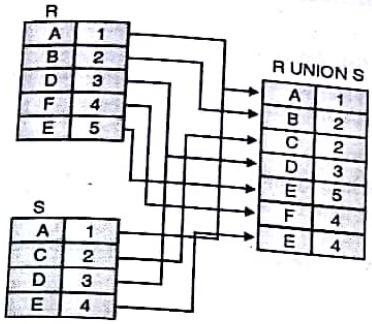


Fig. 4.1 : SET Operation

Example

(i) IT Employee table

Table Name : IT_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23

(ii) Computer department Employee table

Table Name : COMP_Employee		
Eid	**Ename**	**Age**
21	Varsha	24
22	Bhavna	24
23	Geeta	25
24	Amrita	23

Query : Find all Employees in computer and IT departments.

Solution : (IT_Employee) ∪ (COMP_Employee)

Eid	Ename	Age
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23
21	Varsha	24
22	Bhavna	24
23	Geeta	25
24	Amruta	23

Q. 2 Explain Set Intersection operation with suitable examples. May 2014, May 2015

Ans. : Intersect Operator

This operator finds out all rows that are common in table 1 and table 2. If Intersect operator is applied on two queries then it will return all rows that are common in the result of Query 1 and Query 2.

Syntax :

(Query Expression 1) ∩ (Query Expression 2)

R	
A	1
B	2
D	3
F	4
E	5

S	
A	1
C	2
D	3
E	4

R INTERSECTION S	
A	1
D	3

Example

(i) All employees in IT department.

Table Name : IT_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23

(ii) All employees in Vidya Engineering College.

Table Name : Vidya_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
23	Geeta	25
24	Amruta	23
35	Sangita	21

Query : Find all Employees in IT department of Vidya Engineering College.

Solution : (IT_Employee) ∪ (Vidya_Employee)

Eid	Ename	Age
11	Suhas	24
12	Jayendra	24

Q. 3 Explain Set Difference Relational algebra operators with suitable examples. May 2012

Ans. : Difference Operator

This operator finds out all rows that are present in Table 1 and not in table 2. If Intersect operator is applied on two queries then it will return all rows that are present in the result of Query 1 and not in Query 2.

Syntax

(Query Expression 1) - (Query Expression 2)

R	
A	1
B	2
D	3
F	4
E	5

R DIFFERENCE S	
B	2
F	4
E	5

S	
A	1
C	2
D	3
E	4

S DIFFERENCE R	
C	2
E	4

Example

(i) All faculties in IT department of Vidya Engineering College.

Table Name : Vidya_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23

(ii) All faculties in IT department of all colleges.

Table Name : IT_Employee		
Eid	**Ename**	**Age**
11	Suhas	24
12	Jayendra	24
13	Sachin	25
14	Mahesh	23
23	Geeta	25
24	Amruta	23
35	Sangita	21

Query : Find all Employees in IT department but not in Vidya Engineering College.

Solution : (IT_Employee) - (Vidya_Employee)

Eid	Ename	Age
23	Geeta	25
24	Amruta	23
35	Sangita	21

Q. 4 Express Join in terms of basic relational algebra operations.

May 2012, Dec. 2013, May 2014, May 2015

Ans. : Join Operation ($\bowtie_\theta$)

Join operator helps us to retrieve data from multiple tables or relations. Most common type of join is **Natural join** ($\bowtie$) in which column having same name in two table will be taken for joining tables.

Syntax : (< table_name>) $\bowtie_{<join\_condition>}$ (table_name)

There are various types of joins possible in relational algebra.

1. Type 1 : Natural joins
2. Type 2 : Inner joins
3. Type 3 : Outer joins

Type 1 : Natural join ($\bowtie$)

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.

Prerequisites for Natural Join

Both table must have at list one common column with same column name and same data type.

Steps of Working

1. The two table are joined using Cross join
2. DBMS will look for a common column with same name and data type
3. Tuples having exactly same values in common columns are kept in result.

Example

Employee		
Eid	**Ename**	**Did**
1	Amit	10
2	Nitin	30
3	Yogesh	50

Department	
Did	**Dname**
10	IT
30	HR
40	TIS

Query : Find all Employees and their respective departments.

Solution : (Employee) $\bowtie$ (Department)

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
Employee Data			Department Data	

In above example the employee data having same did as department data are kept in result set. All non-matching tuples of cross join are ignored.

Type 2 : Inner joins / Theta Join ($\bowtie_\theta$)

Theta join will combines tuples from multiple relations if they satisfy the specified join condition. This join condition is also called as Theta and denoted by the symbol 0. The tables are joined according to join conditions. The only rows with matching values are combined using inner join. Inner join will ignore all tuple does not find matching tuple in other table.

Example :

Query : Find all Employees and their respective departments.

Solution : Employee $\bowtie_{employee.did = Department.did}$ Department

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
Employee Data	Department Data			

In above example the employee data having did exactly same as department data are kept in result set. All non-matching tuples of cross join are ignored.

Type 3 : Outer joins

In an inner join or in case of a simple join, the resultant table contains only the combinations of rows that satisfy the join conditions. Rows that do not satisfy the join conditions are discarded. Outer join, joins two table although there is no match between two joining tables. Outer joins are useful when you are trying to determine which values in related tables cause referential

integrity problem. Such problems are created when foreign key values do not match the primary key values in related table.

(i) **Left outer join :** Table on left side of operator may contain null values. Left outer join takes all tuples in the left relation that did not match with any tuple in the right relation.

Example

Query : Find all Employees and their respective department data.

Solution : Employee $\bowtie_{=}$ $_{\text{employee.did= Department.did}}$ Department

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
3	Yogesh	50	Null	Null
Employee Data			Department Data	

In above example the employee data having did exactly same as department data are kept in result set. All left side non-matching tuples of cross join are also considered.

(ii) **Right outer join**

Table on right side of operator may contain null values. Right outer join takes all tuples in the right relation that did not match with any tuple in the left relation.

Example

Query : Find all departments with employee data.

Solution : Employee $=\bowtie_{\text{employee.did= Department.did}}$ Department

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
Null	Null	Null	40	TIS
Employee Data			Department Data	

In above example the employee data having did exactly same as department data are kept in result set. All right side non-matching tuples of cross join are also considered.

(iii) **Full outer join :** Any table on both sides of operator may contain null values.

Example

Query : Find all Employees and departments.

Solution : Employee $=\bowtie_{=}$ $_{\text{employee.did= Department.did}}$ Department

Eid	Ename	Did	Did	Dname
1	Amit	10	10	IT
2	Nitin	30	30	HR
3	Yogesh	50	Null	Null
Null	Null	Null	40	TIS
Employee Data			Department Data	

In above example the employee data having did exactly same as department data are kept in result set. All right side non-matching tuples of cross join are also considered.

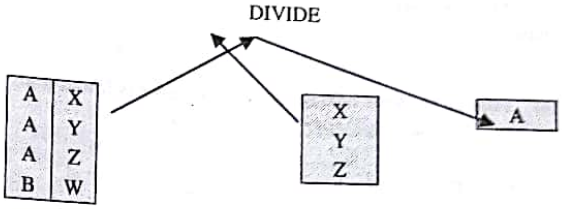
Q. 5 Explain Division Relational algebra operators with suitable examples.

May 2012, Dec. 2013, May 2015

Ans. : Relational Division Operator

The divide operator operates on two tables that must have common columns between them. The relational divide operator (so called to distinguish it from mathematical division) returns the records in one record set that have values that match all the corresponding values in the second record set. The output of divide operation is one column in which values of common column in both tables match.

Syntax : (Query Expression 1) ÷ (Query Expression 2)



DIVIDE Operator

Example

Student Table			
Stud_ID	Sname	Course_ID	Gender
1	Mahesh	100	M
2	Manish	100	M
3	Amruta	100	F
3	Amruta	200	F
6	Neesha	100	F
3	Amruta	300	F
6	Neesha	300	F
6	Neesha	200	F

Course Table contains all courses that trainer 401 is taking.

Course_ID	Trainer_ID
100	401
200	401
300	401

Find female students take ALL the courses that 401 are taking.

Student ÷ Course
OR

$\pi_{\text{s.Stud\_ID, s.Sname, s.Gender, s.Course\_ID}}(\rho_s(\text{Student}) \div \rho_c(\text{Course}))$

s.Stud_ID	s.Sname	s.Gender	s.Course_ID
3	Amruta	F	200
6	Neesha	F	100

Q. 6 Consider the following relations for database that keeps track of student enrollment in courses and books issued for each course.

STUDENT (Ssn, Name, Subject, DOB)
COURSE (Course_id, Name, Dept)
ENROLL (Ssn, Course_id, Semester, Grade)
Book_issued (Course_Id, Semester, ISBN)
TEXT (ISBN, Title, Publisher, Author)
Write any 5 Queries in relational algebra.

Ans. :

(1) Write a query to select all courses available in institute.

$\Pi_{course\_id, CName, Dept}$ (COURSE)

(2) Find all student details registered for course id 10.

$\Pi_{Ssn, Name}$ ($\sigma_{course\_id = 10}$(ENROLL ⋈ STUDENT))

(3) Find various book titles and authors for semester higher than 3.

$\Pi_{ISBN, Title, Author}$ ($\sigma_{semester > 3}$(Book_Issed ⋈ TEXT))

(4) Find all students belongs to IT Depatment (without join)

a) To find course_id of 'IT' Department

$T_1 \leftarrow \Pi_{course\_id}$ ($\sigma_{Dept = 'IT'}$ (COURSE))

b) To find all students enrolled for above course id.

$T_2 \leftarrow \Pi_{ssn}$, (ENROLL ⋈ T_1)

c) To find student details having above Ssn.

Ans. : $\leftarrow \Pi_{Ssn, Name, DOB}$ (STUDENT ⋈ T_2)

Q. 7 Consider the relations given below :

Dealer (Dealer-no, DealerName, address)

Part (Part-no, Part-name, color)

Assigned-to (Dealer-no, Part-no, cost)

Give an expression in relational algebra the following queries :

(i) Find the name of all dealers who supply 'Red' Parts.

(ii) Find the name of the dealers who supply both Yellow and Green Parts.

(iii) Find the name of the dealers who supply all the Parts.

(iv) List all dealer names.

Ans. :

(i) The name of all dealers who supply 'Red' Parts..

$\Pi_{Dealername}$ ($\sigma_{course='red'}$ (Dealer ⋈ Part))

(ii) The name of the dealers who supply both Yellow and Green Parts

$\Pi_{Dealername}$ ($\sigma_{course='red' \ OR \ course='yellow'}$ (Dealer ⋈ Part))

(iii) The name of the dealers who supply all the Parts.

$\Pi_{Dealername}$ (Dealer ⋈ Part))

(iv) The list of all dealer names

$\Pi_{Dealername}$ (Dealer)

Chapter 5 : Structured Query Language

Q. 1 Explain role SQL with example.

Ans. : Role of SQL : SQL (Structured Query Language) is a computer language aimed to store, manipulate, and retrieve data stored in relational databases. SQL is a keyword based language and each statement begins with a unique keyword.

SQL syntax is not case sensitive.

1) SQL is an interactive query language which can be used to retrieve data from database.
2) SQL is a database programming language which can be used along with programming language to access data from database.
3) SQL is a database administration language which can be used to monitor and control data access by various users.
4) SQL can be used as an Internet data access language.

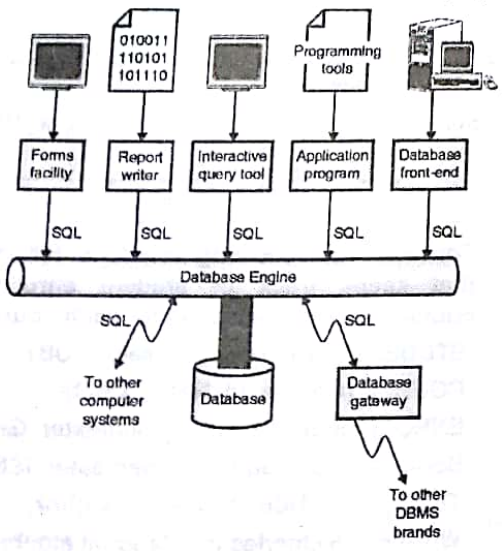


Fig. 5.1 : Role of SQL in DBMS

Q. 2 Explain various SQL data types with example.

Ans. : SQL Data Types

The basic data types available with SQL standard are as enlisted below, all data types may not be supported by SQL server or Oracle.

Data types include numeric, character string, bit string, Boolean and date time.

1. Numeric data types

This datatype is used to store a number values that can be decimal or floating point values.

(a) Integer number of various size : These types of system are used to store natural numbers which are not having any decimal values.

Example : 111, 23 etc.

As per size of number use following types of integers.

(i) INTEGER (p) (ii) INTEGER or INT

(iii) SMALLINT (iv) BIGINT

(b) Floating point numbers of various precision : This system used for storing decimal numbers which may be of greater size than integers.

Example : 11.2, 12.3 etc.

As per size of floating number use following types of numbers.

(i) FLOAT or REAL (ii) DOUBLE PRECISION

(c) Formaed numbers : This system used for storing some special numbers which may be of greater size than integers and floating point numbers.

Example : 1.12342 (Numeric(1,5)),12.234 (Numeric(2,3)) etc.

(i) DECIMAL or DEC (i, j)

(ii) NUMERIC (i, j)

Where decimal i = Precision = Total number of digits after point
j = Scale = Total number of digits after decimal point.
(default value is 0)

2. Character string data type

This data type is used to store a character string which is combination of some alpha bates and enclosed in single quotation marks.

Example : 'Mahesh','abc' etc.

(a) Fixed length : CHAR (n), Where n = number of characters

Example : If 'abc' is stored in char (10) will be stored as 'abc '.

(abc padded with 7 blank spaces)

(b) Varying length : VARCHAR (n) Where n = maximum number of characters

Example : If 'abc' is stored in VARCHAR (10) will be stored as 'abc' (no balnk spaces)

3. Date time data type

(a) Date : The DATE data type has ten positions, and its components are YEAR, MONTH and DAY in form YYYY-MM-DD. The length is 10. Generally not supported by SQL server.(Supported by DB2)

Example : Date '2009-01-01' (as 'YYYY-MM-DD')

(b) Time : The TIME data type has at least eight positions, and its components are HOUR, MINUTES and SECOND in form HH:MM:SS [.sF] where F is the fractional part of the SECOND value. Generally not supported by SQL server. If a second's precision is not specified, *s* defaults to 0. The length is 8 (or 9 + *s*, if $s > 0$).

Example : Time '11:16:59' (as 'HH:MM:SS')

(c) Timestamp / date time : The TIMESTAMP data type includes both date and time fields, plus a minimum of six positions and for decimal fractions of second and optimal with TIMEZONE Qualifier. Represented using the fields YEAR, MONTH, DAY, HOUR, MINUTE and SECOND in the format YYYY-MM-DD HH:MM:SS[.sF] where F is the fractional part of the SECOND value. If a second precision is not specified, *s* defaults to 6. The length is 26 (or 19, if s = 0 or 20+*s*, if $s > 0$).

Example : Timestamp '2009:01:01 11:16:59 648302' (as 'YYYY-MM-DD HH:MM:SS TIMEZONE')

CurrentTimeStamp : Local date and time without time zone

(d) Interval : This specifies an interval a relative value that can be used to increment or decrement an absolute value of date, time or timestamp.

INTERVAL YEAR TO MONTH Datatype

Example : '21-5' Year(2) To Month indicates an interval of 21 years and 5 months '21-5' Year(2) indicates an interval of 21 Years. '5' Month(2) indicates an interval of 5 month.

INTERVAL DAY TO SECOND Datatype

Example : '5 03:15:20' Day(2) To Second indicates an interval of 5 days,3 hours,15 minutes and 20 seconds. Generally not supported by SQL server but supported by Oracle.

Q. 3 Explain DDL commands with example. May 2017

Ans. : Data Definition Language (DDL)

To create database schema and database objects like table **Data Definition Language (DDL)** can be used. DDL statements are used to build and modify the structure of your tables and other objects in the database.

The set of DDL commands are as below,

1. CREATE Statement : To create Database objects
2. ALTER Statement : To modify structure of database objects
3. DROP Statement : To remove database objects
4. RENAME Statement : To Rename Database objects
5. TRUNCATE Statement : To empty the database table

When you execute a DDL statement, it takes effect immediately, as it is **Autocommited** into database. Hence no rollback operation (Undo) can be performed with these set of commands.

1. CREATE Statement / CREATE Table : CREATE statement is used to create new database objects like table, index and others. CREATE TABLE is the command in database system is used to create a new table with unique name or identifier.

This statement used to create database object.

Syntax :

```
CREATE TABLE <Table_Name>
  ( Column_1 datatype,
  Column_2 datatype,
  ....
  Column_n datatype
    );
```

Example

```
SQL> CREATE TABLE Employee
    (   Eid  INT,
  Name VARCHAR (20),
  Age   INT,
  Address    CHAR (25),
  Salary DECIMAL (18, 2)
        );
Query OK, 0 rows affected (0.01 sec)
```

2. Alter Table : Once database object is created in database, may require to the ALTER command is used to update structure of database object. The ALTER TABLE statement can be used to add, delete, or modify columns in an existing table. The ALTER TABLE command can also be used to add and drop various constraints on an existing table.

Syntax :

```
ALTER TABLE <Table_Name>
  ADD Column_1 datatype;

ALTER TABLE <Table_Name>
  Modify Column_1 New_datatype;

ALTER TABLE <Table_Name>
  DROP Column_1;
```

Example

```
SQL> ALTER TABLE Employee
     ADD Address VARCHAR (100);
Query OK, 0 rows affected (0.01 sec)
```

3. **Rename Table :** It is possible to change name of table with or without data in it using simple RENAME command. Rename any table object at any point of time.

Syntax :

```
RENAME TABLE <Table_Name> To <New_Table_Name>;
```

Example :

```
SQL> RENAME TABLE Employee To EMP;
```

4. **Truncate Table :** The SQL TRUNCATE TABLE command is used to delete all data from an existing table. It is possible to da same action with DROP TABLE command but it would remove complete table structure form the database.

A DELETE command will also remove all data from table but with DELETE data deletion can be rolled back and truncate acts as permanent data deletion with no roll back possible. If any delete triggers are defined on the table, then the triggers are not fired on truncate table. Truncate will de-allocates memory space. So that the free space can be used by other tables unlike DELETE command.

Syntax

```
TRUNCATE TABLE <Table_Name>;
```

Example

```
SQL> TRUNCATE TABLE EMP;
```

5. ***Drop Command / DROP Table***

Drop command can be used to remove database any objects from user database. The SQL DROP TABLE statement is used to remove a table definition and all related data like indexes, triggers, constraints and permission specifications for that table. The developer must be careful while running this command because once a table is dropped then all the information available in that table will also be lost forever and no roll back can be done.

Syntax

```
DROP TABLE <Table_Name>;
```

Example :

If we want to permanently remove the Employee table that we created, we'd use the following command

```
SQL> DROP TABLE Employee;
Query OK, 0 rows affected (0.01 sec)
```

Q. 4 Explain DML commands with syntax. May 2017

Ans. : DML commands

Data Manipulation Language (DML) statements are used for manipulating or managing data in database. DML commands are not auto-committed like DDL statements. It means changes done by DML command can be rolled back. Or in other words the DML statements do not implicitly commit the current transaction.

DML is set of commands used to,

1. **INSERT Statement :** Insert statement used to add records to the existing table. To insert data into a table, SQL INSERT INTO command can be used. To insert few values in table as per columns names we can use generic syntax as below,

```
INSERT INTO <Table_Name> (Column1, . . . , ColumnN)
VALUES (column1. . . , columnN);
```

If all values for all the columns of the table are to be added then also no need to specify the column names in the SQL query. But, we need to make sure the order of the values is in the same order as the columns in the table.

```
INSERT INTO <Table_Name>
VALUES (column1. . . , columnN);
```

Example

```
SQL> INSERT INTO Employee VALUES (1001, 'Mahesh');
SQL> INSERT INTO Employee VALUES (NULL, 'Jayendra');

SQL> INSERT INTO Employee (Name, Eid) VALUES
('Sachin' , 1002);
SQL> INSERT INTO Employee (Name, Eid) VALUES ('Suhas'
, NULL);
```

2. **DELETE Statement :** Delete statement is used to delete some or all records from the existing table. To delete data into a table, SQL DELETE command can be used. To delete all rows in table we can use generic syntax as below,

Syntax :

```
DELETE
FROM <Table_Name>;
```

To delete selected rows from table we can specify the WHERE condition.

```
DELETE
FROM <Table_Name>
WHERE <Condition>;
```

Example

```
DELETE
FROM    Employee
WHERE   Eid IS NULL;
```

3. **UPDATE Statement :** The UPDATE statement is used to modify the existing data present in a table. To update data in a table, SQL UPDATE command can be used. To update all rows in table we can use generic syntax as below,

```
UPDATE  <Table_Name>
SET     column1 = new_value;
```

To update selected rows from table we can specify the WHERE condition in Update statement.

```
UPDATE  <Table_Name>
SET     column1=new_value
WHERE   condition;
```

Example

```
SQL>    UPDATE  Employee
        SET     Eid=1002
        WHERE   name = 'Suhas';
```

Q. 5 Write a note on DCL.

Ans. : Data Control Language (DCL)

Data Control Language (DCL) is used to control various user actions (or privileges) in Database. To perform any operation in the database user needs **privileges** like creating tables, sequences or views.

DCL is set of commands used to,

1. **Grant** : Gives some privilege to user for performing task on database.
2. **Revoke** : Take back permissions given from user.

Privileges can be of many types,

a. **System Privileges** : creating a table is types of system privilege.

b. **Object Privileges** : To execute query on tables object privilege can be used.

c. **Ownership Privileges** : To execute query on tables created by same user.

Q. 6 Enlist the privileges with suitable example.

Ans. : Privileges

The set of actions that a user can perform on a database object are called the privileges. Privilege is right to execute particular SQL statement on database. The high level user (Like DBA) has power to grant access to database and its object.

1. **System privileges** : System privileges are rights and restriction that are implemented on databases to control which users can access how much data in the database. User requires system privileges to gain access to database. System privileges are generally provided by DBA. Few system privileges are as below,

System privileges	Authorized to
CREATE USER	Create number of users in DBMS
DROP USER	Drop any other users in DBMS
CREATE ANY TABLE	Create table object in any schema.
SELECT ANY TABLE	Query table object or view in any schema.
DROP ANY TABLE	Drop table object in any schema.

2. **Object privileges** : Object privileges are rights and restrictions to change contents of database objects. User requires object privileges to manipulate the content of object within database. Once created object in a database, after some time there may be few changes needs to be introduced in object.

Not all database users are allowed to make such changes in database; hence administrator should have control over all objects modification. The user which has GRANT ANY PRIVILEGE system privilege granted to him then he can act like administrator to control database modifications. Different objects has different privileges assigned for him,

Few object privileges are as below,

Object privileges	Authorized to
SELECT	Select rows from table or view
INSERT	Add new rows to table or view
DELETE	Remove some rows from table or view
UPDATE	Modify content of rows from table or view
EXECUTE	To run procedure
REFERENCES	To reference a particular table using foreign key and check constraint

3. **Ownership privileges** : Whenever create a database object (like table or view) with the CREATE statement, will become its owner and get full privileges for the table. (Like SELECT, INSERT, DELETE, UPDATE, and all other privileges). All other users are having no privileges on the newly created database object. Owner of database object can explicitly give grant privileges to any other user by using the GRANT statement. Whenever user creating a view with the CREATE VIEW statement, user become the owner of that view, but you do not necessarily receive all privileges as you require the SELECT privilege on each of base tables on which view is defined.

Q. 7 Write syntax for GRANT privileges.

Ans. : GRANT privileges

A system privilege is the right to perform a particular action, or to perform an action on any schema objects of a particular type. An authorized user may pass on this authorization to other users. This process is called as granting of privileges. Generally GRANT statement is used by owner of table or view to give other users access permissions. In SQL user accounts must present in system before grant privileges to him.

Syntax

```
Grant <ALL | privilege list>
ON <relation name or view name>
TO <user | role list | PUBLIC >
   [WITH GRANT OPTION]
```

Privilege list	Meaning
ALTER	Table and views
CREATE	Table and views
DROP	Table and views
DELETE	Tables and views
INSERT	Tables and views
SELECT	Tables and views
UPDATE	Tables and views
ALL	Tables and views

Q. 8 Write a short note on Revoking of privileges.

Ans. : Revoking of Privileges : Reject the privileges given to particular user with help of revoke statement. To revoke an authorization, use the **revoke** statement.

Syntax

```
REVOKE   <ALL | privilege list>
ON       <relation name or view name>
FROM <user | role list | PUBLIC>
[RESTRICT/ CASCADE]
```

CASCADE : will revoke all privileges along with all dependent grant privileges

RESTRICT : This will not revoke all related grants only removes that GRANT only.

Examples

The revocation of privileges from user or role may cause other user or roles also have to leave that privilege. This behaviour is called cascading of the revoke.

(a) To remove select privilege from users U1, U2 and U3.

```
REVOKE    SELECT
ON    mydb.mytbl
FROM 'mahesh'@'somehost';
```

(b) To remove update rights on amount column of Emp_Salary from U1, U2 and U3.

```
REVOKE    UPDATE (amount)
ON    Emp_Salary
FROM 'mahesh'@'somehost';
```

(c) To remove reference right on amount column from user U1.

```
REVOKE    REFERENCES (amount)
ON    Emp_Salary
FROM 'mahesh'@'somehost';
```

The revoke statements may alternatively specify restrict if don't want cascade behavior.

```
REVOKE    SELECT
ON    Emp_Salary
FROM 'mahesh'@'somehost'
RESTRICT;
```

Q. 9 For the given database, write SQL queries.

Employee (Eid. Name, Street, City)
Works (Eid. Cid. salary)
Manager (Eid, Manager_Name)
Company(Cid, Company_name, city)

(i) Modify the database so that 'Jack' now lives in 'Newyork':.

(ii) Give all employees of 'ANZ corporation' a 10% raise in salary :

Ans. :

(i) Modify the database so that 'Jack' now lives in 'Newyork':

```
MySQL>  UPDATE  Employee
  SET    City = 'Newyork'
  WHERE      Name = 'Jack';
```

(ii) Give all employees of 'ANZ corporation' a 10% raise in salary :

```
MySQL>  UPDATE  Works
  SET    Salary = (salary+(0.1*salary))
  WHERE      CID IN ( SELECT    Cid
              FROM     Company
              WHERE    Company_name= 'ANZ
             corporation');
```

Q. 10 Consider insurance database given below and answer the following queries in SQL.

Person (driver_id, name, address)
Car (license, model, year)
Accident (report_no, adate, location)
Owns (driver_id, license)
Participated (driver_id, license, report_no, damage_amount). Dec. 2014

Ans. :

(i) Add new accident to database.

```
MySQL> INSERT INTO Accident (report_no, adate,
location)
  VALUES ('111', '01/01/2014','Pune');
```

(ii) Delete `Santro' belonging to 'John Smith'.

```
MySQL>  DELETE
  FROM  CAR
  WHERE      Model = 'SANTRO'
    AND
    License IN ( SELECT license
            FROM   Owns
            WHERE  Driver_id IN ( SELECT driver_id
                        FROM person
                        WHERE name = 'John Smith')
            );
```

Q. 11 Consider the following employee database.

Employee (empname, street, city, date_of_joining)
Works (empname, company _name, salary)
Company (company_name, city)
Manages (empname, manager_name).
Write SQL queries for the following statements :

(i) Modify the database so that 'John' now lives in 'Mumbai'.

(ii) Give all employees of ABC Corporation' a 10% raise. Dec. 2013

Ans. :

(i) Modify the database so that 'John' now lives in 'Mumbai'.

```
MySQL>  UPDATE  Employee
  SET    City= 'Mumbai'
  WHERE      Empname= 'JOHN';
```

(ii) Give all employees of ABC Corporation' a 10% raise.

```
MySQL>  UPDATE  Works
  SET    Salary=0.1*salary
  WHERE      Company_name= 'ABC Corporation';
```

Q. 12 Employees (Empid, Fname, Lname, Email, Phoneno, Hiredate, Jobid, Salary, Mid, Did)

Departments (Did, Dname, Managerid, Locationid)

Locations (Locationid, Streetadd, Postalcode, City)

Write the SQL queries for the following :

1. **List the employees have a manager who works for a department based in the U.S.**
2. **Write a query to display the details of all employees in the Finance department.**
3. **Give 10% hike to all the employees working in Did 20.**

4. **Write a query to display all the information of the employees whose salary is within the range 1000 and 3000.**
5. **Display the information of all the employees whose first name starts with 'R' in descending order of their salary.**

Ans. :

1. List the employees have a manager who works for a department based in the U.S.
```
SELECT *
FROM Employees e
    INNER JOIN Departments d ON e.did = d.did
    INNER JOIN Locations l ON l.locationid=d.locationid
WHERE City ='US';
```
2. Write a query to display the details of all employees in the Finance department.
```
SELECT *
FROM Employees e
    INNER JOIN Departments d ON e.did = d.did
WHERE Dname='Finance';
```
3. Give 10% hike to all the employees working in Did 20.
```
UPDATE employees
SET Salary = 1.1* Salary
WHERE did =20;
```
4. Write a query to display all the information of the employees whose salary is within the range 1000 and 3000.
```
SELECT *
FROM employees
WHERE Salary BETWEEN 1000 AND 3000;
```
5. Display the information of all the employees whose first name starts with R' in descending order of their salary.
```
SELECT *
FROM employees
WHERE Fname LIKE 'R%'
    ORDER BY Salary DESC;
```

Chapter 6 : Display Data using SQL

Q. 1 Explain the SELECT and FROM clause of SQL queries with appropriate example.

Ans. :

The SELECT statement in SQL is used to select data and display from a table in database. The SELECT can also include SQL built in function to manipulate the data in database just for display purpose like aggregate functions. The SELECT can be used for displaying various attributes or columns of a table.

```
SELECT    [DISTINCT] [ * /Column1,Column2...]
FROM <Table_Name>;
```

Display all data in table

SELECT is basic statement used to retrieve all or some columns of data from table. Select all columns from table by specify * as column name.

Syntax

```
SQL>SELECT  *
  FROM <Table_Name>;
```

Example

Select all details of employees in company.

```
MYSQL> SELECT *
  FROM Employee;
+----+---------+-----+---------+---------+
| ID | NAME    | AGE | ADDRESS | SALARY  |
+----+---------+-----+---------+---------+
|  1 | Yogesh  |  32 | Mumbai  | 12000.00 |
|  2 | Karan   |  25 | Nagar   |  2500.00 |
|  3 | Akshay  |  43 | Delhi   |  4000.00 |
|  4 | Chetan  |  25 | Mumbai  |  6800.00 |
|  5 | Hardik  |  37 | Mumbai  |  8300.00 |
|  6 | Kamal   |  32 | Pune    |  4100.00 |
|  7 | Mahesh  |  24 | Pune    | 15000.00 |
+----+---------+-----+---------+---------+
7 rows in set (0.00 sec)
```

Display limited column data in table

SELECT statement can be used to retrieve specific columns of data from table to maintain security of data. Select columns from table specifying names of required columns.

Syntax :

```
SELECT    Column1, Column2
FROM <Table_Name>;
```

Example : Select Id and names of employees in company.

```
MYSQL> SELECT ID, NAME
  FROM Employee;
+----+----------+
| 1 | Yogesh  |
| 2 | Karan   -+----------+
| ID | NAME    |
|
| 3 | Akshay  |
| 4 | Chetan  |
| 5 | Hardik  |
| 6 | Kamal   |
| 7 | Mahesh  |
+----+----------+
7 rows in set (0.01 sec)
```

Display unique records in table : SELECT is basic statement used to retrieve all or some columns of data from table as explained in above examples. SELECT can have two options as SELECT ALL records in which query result including duplicate records along with unique records or SELECT DISTINCT records to show only unique records. SELECT ALL is default and SELECT DISTNCT will search for distinct rows of in table. Either select all columns from table by specify * as column name or we can specify the required name of columns. In any case DISTINCT will look for unique record set for display purpose only.

Syntax

```
SELECT    DISTINCT [*/Column1,Column2...]
FROM <Table_Name>;
```

Example

Select different locations of above employees.

```
MYSQL> SELECT DISTINCT ADDRESS
   FROM  Employee;

+-----------+
| ADDRESS   |
+-----------+
| Mumbai    |
| Nagar     |
| Delhi     |
| Pune      |
+-----------+
4 rows in set (0.04 sec)
```

Q. 2 Explain 'ORDER BY' clause.

Ans. : 'ORDER BY' clause

The ORDER BY keyword is used to sort the result-set by a specified column. The ORDER BY keyword sorts the records in ascending order by default. Sort the records in a descending order, use the DESC keyword.

Syntax

```
SELECT    column_name
FROM      table_name
ORDER BY  column_name ASC|DESC
```

Example

Select details of employees in company in alphabetical order of names.

```
MYSQL> SELECT*
   FROM      Employee
   ORDER BY  NAME;

+----+---------+-----+-----------+---------+
| ID | NAME    | AGE | ADDRESS   | SALARY  |
+----+---------+-----+-----------+---------+
|  3 | Akshay  |  43 | Delhi     | 4000.00 |
|  4 | Chetan  |  25 | Mumbai    | 6800.00 |
|  5 | Hardik  |  37 | Mumbai    | 8300.00 |
|  6 | Kamal   |  32 | Pune      | 4100.00 |
|  2 | Karan   |  25 | Nagar     | 2500.00 |
|  7 | Mahesh  |  24 | Pune      | 15000.00 |
|  1 | Yogesh  |  32 | Mumbai    | 12000.00 |
+----+---------+-----+-----------+---------+
7 rows in set (0.01 sec)
```

Now select all employees from the table above, however, sort the employees by reverse order of their names. n second query ORDER BY 2 indicates order by second column of table.

```
MYSQL> SELECT  *
   FROM       Employee
   ORDER BY  NAME DESC;
OR
MYSQL> SELECT  *
   FROM       Employee
   ORDER BY  2 DESC;

+----+---------+-----+-----------+---------+
| ID | Name    | Age | Address   | Salary  |
+----+---------+-----+-----------+---------+
|  1 | Yogesh  |  32 | Mumbai    | 12000.00 |
|  7 | Mahesh  |  24 | Pune      | 15000.00 |
|  2 | Karan   |  25 | Nagar     | 2500.00 |
|  6 | Kamal   |  32 | Pune      | 4100.00 |
|  5 | Hardik  |  37 | Mumbai    | 8300.00 |
|  4 | Chetan  |  25 | Mumbai    | 6800.00 |
|  3 | Akshay  |  43 | Delhi     | 4000.00 |
+----+---------+-----+-----------+---------+
7 rows in set (0.00 sec)
```

The ordering can be done on numeric column like salary by default ascending OR descending ordering.

```
MYSQL> SELECT  *
   FROM Employee
   ORDER BY SALARY DESC;
OR
MYSQL> SELECT  *
   FROM Employee
   ORDER BY 5 DESC;

+----+---------+-----+-----------+---------+
| ID | Name    | Age | Address   | Salary  |
+----+---------+-----+-----------+---------+
|  7 | Mahesh  |  24 | Pune      | 15000.00 |
|  1 | Yogesh  |  32 | Mumbai    | 12000.00 |
|  5 | Hardik  |  37 | Mumbai    | 8300.00 |
|  4 | Chetan  |  25 | Mumbai    | 6800.00 |
|  6 | Kamal   |  32 | Pune      | 4100.00 |
|  3 | Akshay  |  43 | Delhi     | 4000.00 |
|  2 | Karan   |  25 | Nagar     | 2500.00 |
+----+---------+-----+-----------+---------+
7 rows in set (0.00 sec)
```

Q. 3 Explain the Where clause of SQL queries with appropriate example.

Ans. : Where clause

The SQL can print selected rows by applying selection operation in SQL with help of 'WHERE' Clause. WHERE statement is an optional statement. The condition written in WHERE clause is a row qualifier condition that row must satisfy to print that row in final result of query. The query without WHERE condition will print all rows by default.

Syntax

```
SELECT    *
FROM      Table_Name
[WHERE  condition]
```

Examples

Select details of employees in company having salary above 5000.

```
MYSQL> SELECT  *
```

```
FROM Employee
WHERE SALARY > 5000
Order By ID;
+----+---------+-----+---------+---------+
| ID | NAME    | AGE | ADDRESS | SALARY  |
+----+---------+-----+---------+---------+
|  1 | Yogesh  |  32 | Mumbai  | 12000.00 |
|  4 | Chetan  |  25 | Mumbai  |  6800.00 |
|  5 | Hardik  |  37 | Mumbai  |  8300.00 |
|  7 | Mahesh  |  24 | Pune    | 15000.00 |
+----+---------+-----+---------+---------+
4 rows in set (0.01 sec)
```

Q. 4 Explain Aggregate function with example. May 2016, Dec. 2016

Ans. : Aggregate Functions

The management report require lot of decision making data to get summarized information from table like average, sum, minimum. SQL provides various aggregate functions which can summarize data of given table. The function operates on the table data produces a single output. Such queries are generally used for producing reports and summary forms in an application. The aggregate data functions ignore all null value in column on which aggregate data function if applied.

Types of aggregate functions

a) **COUNT ([DISTINCT] C) :** The number of (unique) values in the column C.

b) **SUM ([DISTINCT] C) :** The sum of all (unique) values in the column C.

c) **AVG ([DISTINCT] C) :** The average of all (unique) values in the column C.

d) **MIN (C) :** The minimum value in the column C.

e) **MAX (C) :** The maximum value in the column C.

Example

Consider the student table indicate assessment marks of every student. The blanks in mark indicates absent student in exam. Assessment is conducted out of 100 marks. If school management want to analyse performance of class using summarized data.

ID	Name	Marks	Department	Subject	Grant
1	Mahesh	90	IT	DB	5000
2	Suhas	60	CS	DB	
3	Jayendra	89	IT	DB	3000
4	Sachin	99	CS	DB	7000
5	Vishal	78	IT	DB	
6	Payal	90	MECH	CPP	3000
7	Kasturi		MECH	CPP	

(a) COUNT () : This function is used to calculate number of rows (or records) in a table selected by query. Column in the query must be numeric. COUNT returns the number of rows in the table when the column value is not NULL. Count (*) will returns that number of unique tuples present in table.

Find number of students in class.

```
MYSQL> SELECT count (*) Tot_Employees
  FROM      Student;
```

OR

```
MYSQL> SELECT count (ID) Tot_Employees
  FROM      Student;
+---------------+
| Tot_Employees |
+---------------+
|             7 |
+---------------+
1 row in set (0.00 sec)
```

The aggregate data functions ignore all null value in column on which aggregate data function if applied.

Find number of students appeared for exam.

```
MYSQL> SELECT count (Marks)
  FROM      Student;
+--------------+
| count(Marks) |
+--------------+
|            6 |
+--------------+
1 row in set (0.00 sec)
```

(b) SUM () : This function is used to calculate sum of column values in a table selected by query. Column in the query must be numeric. Value of the sum must be within the range of that data type.

Find total of grants given by non IT departments.

```
MYSQL> SELECT SUM(Grants)
  FROM      Student
  WHERE         Department <> 'IT';
+-------------+
| SUM(Grants) |
+-------------+
|   10000.00 |
+-------------+
1 row in set (0.00 sec)
```

(c) AVG () : This function is used to calculate Average of column values in a table selected by query. This function first calculates sum of column and then divide by total number of rows. AVG returns the average of all the values in the specified column. Column in the query must be numeric.

Find average marks scored by non IT departments.

```
MYSQL> SELECT AVG(Marks)
  FROM      Student
  WHERE         Department <> 'IT';
+-------------+
| AVG(Marks) |
+-------------+
|  83.0000 |
+-------------+
1 row in set (0.00 sec)
```

(d) MIN () : This function is used to find minimum value out of column values in a table selected by query.

Column in the query need not be numeric data type.

Find minimum marks scored by non IT departments.

```
MYSQL> SELECT MIN(Marks)
   FROM Student
   WHERE Department <> 'IT';
+------------+
| MIN(Marks) |
+------------+
|         60 |
+------------+
1 row in set (0.00 sec)
```

(e) MAX () : This function is used to find maximum value out of column values in a table selected by query. Column in the query need not be numeric data type.

Find maximum marks scored by non IT departments.

```
MYSQL> SELECT MAX(Marks)
   FROM Student
   WHERE Department <> 'IT';
+------------+
| MAX(Marks) |
+------------+
|         99 |
+------------+
1 row in set (0.00 sec)
```

Q. 5 Explain the term : Group by clause. May 2016

Ans. : Group by clause

The SQL GROUP BY clause is used in with the SQL SELECT statement to arrange data into groups. Related rows can be grouped together by **GROUP BY** clause based on distinct values that exist for specified grouping columns. The GROUP BY statement is used with the SQL aggregate functions to group the data using one or more columns or expression.

The GROUP BY column does not have to be in the SELECT clause. A Grouping Query groups rows based on common values in a set of grouping columns. Rows with the same values for the grouping columns are placed in distinct groups. Each group is treated as a single row in the query result.

Syntax

```
SELECT Column_name
FROM Table_name
[WHERE Condition]
GROUP BY Column_names
[ORDER BY Columns];
```

Example

Consider the student table indicate assessment marks of every student. The blanks in mark indicates absent student in exam. Assessment is conducted out of 100 marks. If school management want to analyse performance of class using summarized data.

ID	Name	Marks	Department	Subject	Grant
1	Mahesh	90	IT	DB	5000
2	Suhas	60	CS	DB	
3	Jayendra	89	IT	DB	3000
4	Sachin	99	CS	DB	7000
5	Vishal	78	IT	DB	
6	Payal	90	MECH	CPP	3000
7	Kasturi		MECH	CPP	

Retrieve all departments in college.

```
MySQL> SELECT Department
   FROM Student
   GROUP BY Department;

+------------+
| Department |
+------------+
| CS         |
| IT         |
| MECH       |
+------------+
3 rows in set (0.00 sec)
```

Q. 6 What do you mean by set operation ?

Ans. :

SET Operations

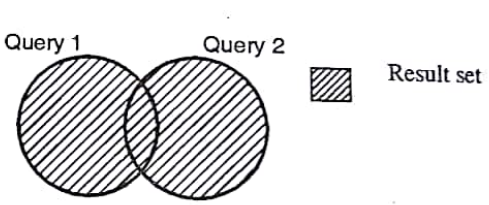
SQL SET operators allow combining results from two or more SELECT statements or combines result set of multiple queries. The results of two queries can be combined using the set operations union, intersection and difference.

Union Operator

Union effectively appends the result of first query to the result of second query. It does not eliminate all duplicate rows and they are printed in result expression.

DISTINCT : Duplicate row shown only once.

ALL : Duplicate row shown only once.



Find all students in all departments.

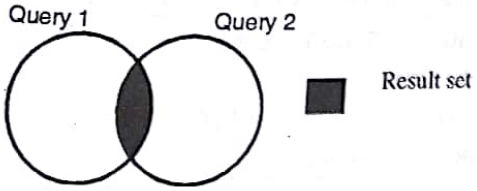
```
MySQL> SELECT *
   FROM student
   WHERE department='IT';
+----+---------+-------+------------+---------+--------+-------------+
| ID | Name    | Marks | Department | Subject | Grants | email       |
+----+---------+-------+------------+---------+--------+-------------+
|  1 | Mahesh  |    90 | IT         | DB      | 5000.00 | a_bh@myc.edu |
|  3 | Jayendra|    89 | IT         | DB      | 3000.00 | a_by@myc.edu |
|  5 | Vishal  |    78 | IT         | DB      |   NULL | is@myc.edu  |
+----+---------+-------+------------+---------+--------+-------------+
3 rows in set (0.00 sec)

MySQL> SELECT * WHERE department='CS';
+----+---------+-------+------------+---------+--------+-------------+
| ID | Name    | Marks | Department | Subject | Grants | email       |
```

```
| 2 | Suhas | 60 | CS | DB | NULL | uh@myc.edu |
| 4 | Sachin | 99 | CS | DB | 7000.00 | a_bc@myc.edu |
2 rows in set (0.00 sec)

MySQL> SELECT *
  FROM student
  WHERE department='MECH';

| ID | Name | Marks | Department | Subject | Grants | email |
| 6 | Payal | 90 | MECH | CPP | 3000.00 | a_by@myc.edu |
| 7 | Kasturi | NULL | MECH | CPP | NULL | a_bs@myc.edu |
2 rows in set (0.00 sec)

/* To Combine Result of above 3 Queries using UNION Operation */

MySQL> SELECT * FROM student WHERE department='IT'
  UNION
  SELECT * FROM student WHERE department='CS'
  UNION
  SELECT * FROM student WHERE department='MECH';

| ID | Name | Marks | Department | Subject | Grants | email |
| 1 | Mahesh | 90 | IT | DB | 5000.00 | a_bh@myc.edu |
| 3 | Jayendra | 89 | IT | DB | 3000.00 | a_by@myc.edu |
| 5 | Vishal | 78 | IT | DB | NULL | is@myc.edu |
| 2 | Suhas | 60 | CS | DB | NULL | uh@myc.edu |
| 4 | Sachin | 99 | CS | DB | 7000.00 | a_bc@myc.edu |
| 6 | Payal | 90 | MECH | CPP | 3000.00 | a_by@myc.edu |
| 7 | Kasturi | NULL | MECH | CPP | NULL | a_bs@myc.edu |
7 rows in set (0.06 sec)
```

Intersect Operator

This operator finds out all rows that are common in both result of Query 1 and in the result of Query 2. It does not eliminate all duplicate rows and they are printed in result expression.

Example

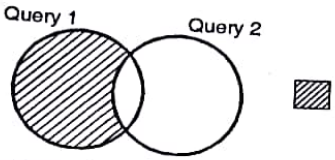


Find all IT students offered grants above 2000.

```
/* Query 1: To print all IT students */
MySQL> SELECT *
  FROM Student
  WHERE Department='IT';

| ID | Name | Marks | Department | Subject | Grants | email |
| 1 | Mahesh | 90 | IT | DB | 5000.00 | a_bh@myc.edu |
| 3 | Jayendra | 89 | IT | DB | 3000.00 | a_by@myc.edu |
| 5 | Vishal | 78 | IT | DB | NULL | is@myc.edu |
3 rows in set (0.00 sec)

/* Query 2: To print all students offered grants above 2000 */

MySQL> SELECT *
  FROM Student
  WHERE Grants>=2000;

| ID | Name | Marks | Department | Subject | Grants | email |
| 1 | Mahesh | 90 | IT | DB | 5000.00 | a_bh@myc.edu |
| 3 | Jayendra | 89 | IT | DB | 3000.00 | a_by@myc.edu |
| 4 | Sachin | 99 | CS | DB | 7000.00 | a_bc@myc.edu |
| 6 | Payal | 90 | MECH | CPP | 3000.00 | a_by@myc.edu |
4 rows in set (0.00 sec)

/* Final Query: To print all IT students offered grants above 2000 */

MySQL> SELECT * FROM Student WHERE Department='IT'
  INTERSECT
  SELECT * FROM Student WHERE Grants>=2000;

| ID | Name | Marks | Department | Subject | Grants | email |
| 1 | Mahesh | 90 | IT | DB | 5000.00 | a_bh@myc.edu |
| 3 | Jayendra | 89 | IT | DB | 3000.00 | a_by@myc.edu |
2 rows in set (0.00 sec)
```

Difference Operator

Returns all rows that are in the result of Query 1 but not in the result of Query 2. Again, duplicates are eliminated unless **ALL** is specified.



Find all non IT students offered grants above 2000.

```
/*Query 1: To print all students offered grants above 2000 */

MySQL>  SELECT  *
    FROM  Student
    WHERE     Grants>=2000;
+----+---------+-------+------------+---------+--------+-----------------+
|ID  | Name    | Marks | Department | Subject | Grants | email           |
+----+---------+-------+------------+---------+--------+-----------------+
|  1 | Mahesh  |    90 | IT         | DB      | 5000.00 | a_bh@myc.edu   |
|  3 | Jayendra|    89 | IT         | DB      | 3000.00 | a_by@myc.edu   |
|  4 | Sachin  |    99 | CS         | DB      | 7000.00 | a_bc@myc.edu   |
|  6 | Payal   |    90 | MECH       | CPP     | 3000.00 | a_by@myc.edu   |
+----+---------+-------+------------+---------+--------+-----------------+
4 rows in set (0.00 sec)

/* Query 2 : To print all IT students */

MySQL>  SELECT  *
    FROM  Student
    WHERE     Department='IT';
+----+---------+-------+------------+---------+--------+-----------------+
| ID | Name    | Marks | Department | Subject | Grants | emai            |
+----+---------+-------+------------+---------+--------+-----------------+
|  1 | Mahesh  |    90 | IT         | DB      | 5000.00 | a_bh@myc.edu   |
|  3 | Jayendra|    89 | IT         | DB      | 3000.00 | a_by@myc.edu   |
|  5 | Vishal  |    78 | IT         | DB      |  NULL   | is@myc.edu     |
+----+---------+-------+------------+---------+--------+-----------------+
3 rows in set (0.00 sec)

/* Final Query : To print all IT students offered grants above
2000 */

MySQL>  SELECT * FROM Student WHERE Grants>=2000
    MINUS
    SELECT * FROM Student WHERE Department='IT';
+----+---------+-------+------------+---------+--------+-----------------+
-+
| ID | Name    | Marks | Department | Subject | Grants | email           |
+----+---------+-------+------------+---------+--------+-----------------+
|  4 | Sachin  |    99 | CS         | DB      | 7000.00 | a_bc@myc.edu   |
|  6 | Payal   |    90 | MECH       | CPP     | 3000.00 | a_by@myc.edu   |
+----+---------+-------+------------+---------+--------+-----------------+
2 rows in set (0.00 sec)
```

Q. 7 **Consider the following relations for database that keeps track of student enrolment in courses and books issued for each course.**

STUDENT (Ssn, Name, Subject, DOB)
COURCE (Course_Id, Name, Dept)
ENROLL (Ssn, Course_Id, Semester, Grade)
BOOK_ISSUED (Course_Id, Semester, ISBN)
TEXT (ISBN, Title, Publisher, Author)
Write a Query to select all courses available in institute.

Ans. :

```
MySQL>  SELECT  *
    FROM COURSE
```

Q. 8 **For the given database, write SQL queries :**

employee (eid, employee_name, street, city)
Works (eid, cid, salary)
Company (cid, company_name, city)
Manager (eid, manager_name)

(i) **Find the names of all employees having "i" as the second letter in their names.**

(ii) **Display the annual salary of all employees.**

Ans. :

(i) Names of all employees having "i" as the second letter in their names.

```
MySQL>  SELECT  Employee_name, street, city
    FROM Employee
    WHERE     Employee_name like '_i%';
```

(ii) Annual salary of all employees.

```
MySQL>  SELECT  Eid, (salary)*12 [annualsalary]
    FROM Works;
```

Q. 9 **Consider the following relations for a book club :**

Members (Member-Id, Name, Designation, Age)
Books (Book-Id, Booktitle, BookAuthor, Bookpublisher, Bookprice)
Reserves (Member-Id, Book-Id, Date)
Find the names of members who are professor older than 50 years. **May 2014**

Ans. :

Names of members who are professor older than 50 years.

```
MySQL>  SELECT  Name
FROM Members
WHERE     Designation LIKE 'Professor'
AND       Age>50;
```

Chapter 7 : Complex Queries

Q. 1 What is JOIN ? Explain different types of JOIN along with example. May 2016

Ans. : JOIN

The relational operations can merge columns from two different tables to form new join table.

A	B	C

D	E

A	B	C	D	E

Fig. 7.1 : Joining operation

Join multiple tables with help of some join condition (Equality or Inequality) or without any join condition (cross join).

The join concepts will be demonstrated using following HR schema which contains Employee table and Department table. These table also having a common column between them.

Cartesian product / Cross join

A cross join performs the relational product or Cartesian product of two tables in such product every tuple of the first table is joined with each tuple in the second table without any join condition. In DBMS, CROSS join occur due to missing WHERE condition in query or some invalid operations in where clause.

A cartesian product is formed when :

1. A join condition is omitted.
2. A join condition is invalid.
3. All rows in the first table are joined to all rows in the second table.

In such cross product operation all tuples are merged, although there is a common column or not and even tuple have matching or non-matching values in order to avoid non-matching tuples, always include a valid join condition in a WHERE clause.

In CROSS joins, every tuple in table 1 will be joined with each tuple in table 2.

Employee Table

Eid	Ename	Did
1	Mahesh	100
2	Suhas	200
3	Jayendra	100

Department Table

Did	Dname
100	HR
200	TIS

Syntax

```
SELECT    Column_List
FROM  Table1
CROSS JOIN Table2
```

Inner join

An INNER JOIN performs the relational product between two tables first and than use join condition to select only tuples satisfying join condition. Inner join will ignore all tuples that does not satisfy join condition or having missing values in it. Inner join will only merge tuples if merged tuples satisfy join condition. Hence it is called as Inner Join.

Syntax

```
SELECT    Column_List
FROM  Table1
  INNER JOIN Table2
  ON (JOIN_Condition);
OR
SELECT    Column_List
FROM  Table1
  INNER JOIN Table2
  ON (JOIN_Condition);
```

Case 1 : Natural join

A natural join will selects all tuples from both participating tables, if and only if both tables has one common column with same attribute value in both tables. In this type of join, joining tables must have at least one common column between them which has same column name and data type.

Working :

Join processor looks for a columns present in both tables. Perform Cross Join between participating tables (unconditional step). Retain row in result if it has matching attribute values in common column.

Syntax

```
SELECT    Column_List
FROM      Table1
      NATURAL JOIN Table2;
```

Case 2 : Self join

A Self-join will work exactly same as Inner join but there is only one participating table and both columns involved in joining are in the same table. The tables being joined in a query do not need to be distinct; join any table to itself. Self-joins are useful for discovering relationships between different columns of the same table. In this type of join query we must use rename operation or table alias.

Case 3 : Non Equi-JOIN

A Non equi-join will work exactly same as Inner join but there is equality join condition is involved in joining participating table. Make use of operators like range operator (BETWEEN) or limit operator (IN).

The join query in which equality condition is not used then such joins are called as **Non Equi-join.**

Outer join

In an inner join, the resultant table contains only the combinations of rows that satisfy the join conditions. Rows that do not satisfy the join conditions are discarded. Outer join, will merge two table although there is no match between attribute values of common column. An outer join makes one of the tables dominant. Such table is called the outer table and other table is called as subordinate table. In an outer join, the resultant table contains the combinations of rows from dominant table that satisfy the join conditions and also rows that do not having matching rows in the subordinate table.

The rows from the dominant table that do not have matching rows in the subordinate table contain NULL values in the columns selected from the subordinate table. The Join that can be used to see rows of table that do not meet the join condition along with rows that satisfies join condition is called as **outer join.**

Syntax

```
SELECT    Column_List
FROM      Table1
```

```
[LEFT/RIGHT/FULL] OUTER JOIN Table2
ON Join_Condition;
```

1. Left outer join

Left outer join has the dominant table of the outer join on left side of the keyword 'outer join'. A left outer join returns all of the rows for which the join condition is true and, in addition, returns all other rows from the dominant table and displays the corresponding values from the subordinate table as NULL.

2. Right outer join

Right outer join has the dominant table of the outer join on right side of the keyword 'outer join'. A right outer join returns all of the rows for which the join condition is true and also includes all other rows from the dominant table and displays the corresponding values from the subordinate table as NULL.

3. Full outer join

In full outer join both the table acts as dominant tables alternatively. A full outer join returns all of the rows for which the join condition is true and also includes,

(a) All other rows from the right table and displays the corresponding values from the left table as NULL.

(b) All other rows from the left table and displays the corresponding values from the right table as NULL.

Q. 2 What is nested sub query ? Explain ANY ALL operators with example.

Ans. : Nested Sub Queries

A Subquery or Nested sub query is a query within another SQL query. Subquery is query appear within **WHERE** or **HAVING** clause of other query.

SELECT	....
FROM	....
WHERE	.... (<,>,<=,>=,<>)
	(Sub Query)
HAVING	.... (IN/ANY/ALL)
	(Sub Query)

Fig. 7.1 : Subquery

Outer query is called as **main query** and inner query which is written in (where or having clause) main query is called **subquery**. **Subquery in WHERE clause :** The result of the subquery (inner query) is used to select some rows from main query. **Subquery in HAVING clause :** The result of the subquery (inner query) is used to select some groups from main query.

Sub queries can be nested within other sub queries.

Syntax

```
SELECT    Select_list
      FROM    Table
      WHERE   expr_operator ( SELECT Select_list
            FROM    Table);
OR
SELECT    Select_list
      FROM    Table
GROUP BY Column
      HAVING  expr_operator
( SELECT Select_list
      FROM    Table);
```

Expr_operator can be of two types like,

(a) Single row operator (<, >, <=, >=, <>)

(b) Multiple row operator (IN, ANY, ALL)

Q. 3 For the following given database, write SQL queries :

person (driver_id #, name, address)
car (license, model, year)
accident (reportCno, date, location)
owns (driver id #, license)
participated (driverid, car, report_number, damage_amount)

(i) Finding the total number of people who owned cars that were involved in an accident in 2007.

(ii) Finding the number of accidents in which the cars belonging to "Ajay" were involved.

Ans. :

(i) Total number of people who owned cars that were involved in an accident in 2007.

```
MySQL> SELECT COUNT(*)
  FROM Person
  WHERE Driver_id IN
  (Select Driverid FROM Participated
    WHERE Car IN
    (Select ReportCno FROM Accident
      WHERE Year(Date) = 2007);
```

(ii) Number of accidents in which the cars belonging to "Ajay" were involved.

```
MySQL> SELECT count(*)
  FROM accident
  WHERE reportCno IN
  (SELECT report_number FROM participated
    WHERE car IN (SELECT driver_id FROM person
          WHERE name = 'Ajay');
```

(iii) Finding the number of accidents that were reported in Mumbai region in the year 2004.

```
MySQL> SELECT COUNT(*)
  FROM accident
  WHERE reportCno IN
  (Select report_number FROM Participated
    WHERE Car IN
    (Select ReportCno FROM Accident
      WHERE location = 'Mumbai'
          AND Year(Date) = 2004);
```

Q. 4 Consider the following relations for database that keeps track of student enrolment in courses and books issued for each course.

STUDENT (Ssn, Name, Subject, DOB)
COURCE (Course_id, Name, Dept)
ENROLL (Ssn, Course_id, Semester, Grade)

BOOK_ISSUED (Course_Id, Semester, ISBN)
TEXT (ISBN, Title, Publisher, Author)

(1) Finding all student details registered for course id 10.

(2) Finding various book titles and authors for semester higher than 3.

(3) Finding all students belongs to IT Department (without join)

(4) Finding total number of student s enrolled in IT Department.

Ans. : (1) All student details registered for course id 10.

```
MySQL> SELECT *
  FROM STUDENT S
  WHERE Ssn = (SELECT Ssn FROM ENROLL E
               WHERE E.Course_id = 10);
```

(2) Various book titles and authors for semester higher than 3.

```
MySQL> SELECT *
  FROM BOOK_ISSUED B
    NATURAL JOIN TEXT T
  WHERE   B.Semester > 3
```

(3) Finding all students belongs to IT Department (without join)

```
MySQL> SELECT Ssn, CName
  FROM STUDENT
  WHERE   Ssn IN (SELECT Ssn FROM ENROLL
            WHERE Course_id IN
              (SELECT Course_id FROM Course
                WHERE   Dept = 'IT'))
```

(4) Finding total number of student s enrolled in IT Department.

```
MySQL> SELECT COUNT (E.Ssn) Total_Students
  FROM STUDENT
    NATURAL JOIN ENROLL
    NATURAL JOIN Course;
```

Q. 5 **Consider the following relations for a book club :**

Members (Member-Id, Name, Designation, Age)
Books (Book-Id, Booktitle, BookAuthor, Bookpublisher, Bookprice)
Reserves (Member-Id, Book-Id, Date)

(i) List the titles of books reserved by professors.

(ii) Find Ids of members who have reserved books that cost more than 500.

(iii) Find the authors and titles of books reserved on 20-09-2012. May 2014

Ans. :

(i) List the titles of books reserved by professors.

```
MySQL> SELECT Booktitle
  FROM Books
  WHERE Book-id IN
    (SELECT Book-id FROM Reserves
      WHERE Member-id IN
        (SELECT Member-id FROM Members
          WHERE Designation='Professor'));
```

(ii) Ids of members who have reserved books that cost more than 500.

```
MySQL> SELECT Member-id FROM Members
  WHERE Member-id IN
    (SELECT Member-id FROM Reserves
      WHERE Book-id IN
        (SELECT Book-id FROM Books
          WHERE Bookprice > 500));
```

(iii) The authors and titles of books reserved on 20-09-2012.

```
MySQL> SELECT Booktitle, BookAuthor
  FROM Books
  WHERE Book-id IN (SELECT Book-id
            FROM Reserves
            WHERE Date = '2012-09-02');
```

Q. 6 **Consider the following employee database.**

Employee (empname, street, city, date_of_joining)
Works (empname, company _name, salary)
Company (company_name, city)
Manages (empname, manager_name).

(i) List all employees who live in the same cities as their managers.

(ii) Find all employees who earn more than average salary of all employees of their company. Dec. 2013

Ans. :

(i) All employees who live in the same cities as their managers.

```
MySQL> SELECT *
  FROM Employee E, Manages M
  WHERE E.empname = M.empname
    AND E.city = M.city);
```

(ii) All employees who earn more than average salary of all employees of their company.

```
MySQL> SELECT   *
  FROM Employee E, Works W
  WHERE E.empname= W.empname
    AND W.salary > (SELECT AVG(salary)
          FROM Employee
          WHERE company_name = W.company_name);
```

Q. 7 **Refer education database mentioned**

Course (course-no, title)
Offering (course-no, off-no, off-date, location)
Teacher (course-no, off-no, emp-no)
Enrolment (course-no, off-no, stud-no, grade)
Employee (emp-no, emp-name, job)
Student (stud-no, stud-name, ph-no)

Write SQL queries for the following :

i) List all the teacher who conduct the course titled Database Systems.

ii) List all the course offered in Thane on 15/8/15.

iii) Find the course enrolled by Monali.

iv) List all the employees who work as Teachers. Dec. 2015

Ans. :

i) All the teacher who conduct the course titled "Database Systems".

```
SELECT  *
FROM    Teacher
WHERE   course-no IN ( SELECT course-no
                       FROM   Course
                       WHERE  title = 'Database Systems')
```

ii) All the course offered in 'Thane' on 15/8/15.

```
SELECT  *
    FROM    Teacher
    WHERE   location LIKE 'Thane'
        AND off-date = '15-08-2015'
```

iii) Course enrolled by "Monali".

```
SELECT  *
  FROM    Course
  WHERE   course-no IN
(SELECT course-no FROM Enrolment WHERE stud-no IN
(SELECT  stud-no  FROM  Student  WHERE  stud-name  =
'Monali'))
```

iv) All the employees who work as Teachers.

```
SELECT  *
FROM    Employee
WHERE   emp-no IN ( SELECT emp-no FROM Teacher )
```

Q. 8 Consider the following employee database

Employee (empname, street, city, date of join)
Works (empname, company-name, salary)
Company (company-name, city)
Manages (empname, manager-name)
Write SQL queries for the following statements :

(i) Find all employees who joined in the month of October.

(ii) Modify the" database so that 'Peter' now lives in 'Newton'.

(iii) List all employees who live in the same cities as their managers.

(iv) Find all employees who earn more than average salary of all employees of their company.

(v) Give all employees of XYZ corporation a 15 percent raise. May 2015

Ans. : (i) insert statement

```
SELECT *
FROM Employee
WHERE MONTH (date of join) = 10;
```

(ii) insert statement

```
UPDATE Employee
SET city= 'Newton'
WHERE empname = 'Peter';
```

(iii) insert statement

```
SELECT *
FROM Employee e
WHERE e. city = (SELECT emp.city
    FROM   Employee emp, Manages  mgr
    WHERE emp.empname=mgr.empname
    AND    emp.empname=e.empname    )
```

(iv) insert statement

```
SELECT   e.empname
FROM     Employee e NATURAL JOIN Works w
WHERE    empname >
(SELECT avg (salary) FROM        Works w WHERE
w.empname=e.empname)
```

(v) Give all employees of XYZ Corporation a 15 percent raise.

```
UPDATE  Employee
SET     salary = 1.15*salary
WHERE       empname > (SELECT empname
            FROM    Works w
        WHERE company-name= 'XYZ corporation')
```

Q. 9 Consider the following employee database

Employee
(emp_name,street,city,date_of_joining)
Works (emp name, company_name, salary)
Company (company_name, city)
Manages (emp _ name,manager_name)
Write SQL queries for following :

(i) Modify the database so that 'Deepa' lives in 'Pune.

(ii) Give all employees of 'XYZ corporation' a 10% rise in salary.

(iii) List all employees who lives in the same city as their companycity.

(iv) Display all employees who joined in the month of 'March'.

(v) Find all employees who earn more than average salary of all employees of their company.

Ans. :

(i) Modify the database so that 'Deepa' lives in 'Pune.

```
Update Employee
Set city ='Pune'
Where emp_name= 'Deepa';
```

(ii) All employees of 'XYZ corporation' a 10% rise in salary.

```
MySQL>  SELECT *
  FROM Employee E, Manages M
  WHERE E.empname = M.empname
    AND E.city = M.city);
```

(iii) All employees who lives in the same city as their companycity.

```
MySQL>  SELECT *
  FROM Employee E, Manages M
  WHERE E.empname = M.empname
    AND E.city = M.city);
```

(iv) All employees who joined in the month of 'March'.

```
SELECT *
```

```
FROM Employee
WHERE month(date_of_joining) = 3;
```

(v) All employees who earn more than average salary of all employees of their company.

```
MySQL> SELECT *
FROM Employee E, Works W
WHERE E.empname= W.empname
  AND W.salary > (SELECT AVG(salary)
                  FROM Employee
                  WHERE company_name = W.company_name);
```

Chapter 8 : SQL Security

Q. 1 What is a view ? How is it created and stored ?

May 2012, May 2015, May 2016, May 2017

Ans. : View : A view is defined as a database object that allows us to create a virtual table in the database whose contents are defined by a query or taken from one or more tables. View is defined to hide complexity of query from user.

View - As a window of entire table : Instead of showing entire table to a user we can show a glimpsed of table to the user which is required for him

Example : Consider a student table contains following columns,

```
STUDENT (Stud_Id, Stud_Name, Std, Div, Addr, Sports, Fees, Cultural_Activity)
```

Now for a sports teacher requires only sports related data of students so we can create view called as Stud_Sports_View for teacher as below which will only depicts sports data of student to sports teacher.

Stud_Sports_View (Stud_Id, Stud_Name, Sports)

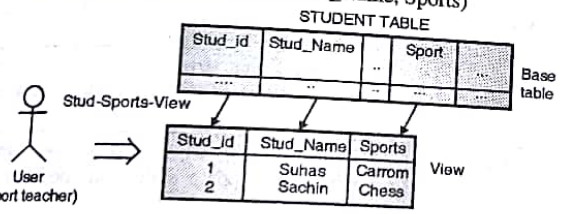


Fig. 8.1 : Overview of view

When we call view in SQL query it refers to database and finds definition of views which is already stored in database. Then the DBMS convert this call of view into equal request on the base tables of the view and carries out the operations written in view definition and returns result set to query from which view is called.

Creating a Views

To create a view a subquery must be embedded within the CREATE VIEW statement.

A simple query is designed and its output can be recorded as a view. The CREATE statement assigns a name to the view and also gives the query which defines the view. To create the view one should have must have privileges to access all of the base tables on which view is defined. Also, the user must have create view permissions from DBA to create a view in the database. The create view can change the name of the column in view as per requirements.

Syntax

```
CREATE [OR REPLACE] VIEW <view name>
AS
SUB QUERY
[WITH CHECK OPTION]
```

OR REPLACE : Change the definition of a view without dropping (ALTER VIEW)

VIEW NAME : Is name given to a view.

SUB QUERY : The query which retrieves the columns of the table that query must have.

WITH CHECK OPTION : This is type of check constraint, which specifies that only those rows which are selected by view can be inserted updated or deleted.

Example

In the college database, we may want to let a Head of Departments see only the FACULTY rows for own department.

```
SQL>CREATE VIEW IT_Faculty
  AS
  SELECT    *
  FROM  FACULTY
  WHERE     Faculty_Dept= 'IT';

/* Selecting Data */

SQL>SELECT    *
  FROM IT_Faculty;
```

Q. 2 What are the types of views ?

Ans. : Types of views

(a) Simple view : The views which are based on only one table called as Simple view. Allow to perform DML (**Data Manipulation Language**) operations with some restrictions. Query defining simple view cannot have any join or grouping condition.

(b) Complex view : The views which are based on more than one table called as complex view. Do not allow DML operations to be performed. Query defining complex view can have join or grouping condition.

Q. 3 How to drop view ? Give Syntax.

Ans. : Drop a view

To drop a view we use DROP VIEW statement. The DROP VIEW statement requires a name to the view. To DROP the view one should have must have privileges to from DBA to DROP a view in database. The DROP view dose not affects base table or any column of base table.

Syntax

```
DROP VIEW <View_name> [RESTRICT|CASCADE]
```

RESTRICT : Delete view only if their in no other view depends on this view.

CASCADE : Delete view along with all dependent view on original view.

Example : Remove a view created in above step.

```
SQL>DROP VIEW JobBelow3K;
View Dropped;
```

Q. 4 What are advantages of Views ?

Ans. : Advantages of Views

(a) Security : Views can restrict user from accessing all data. In case of view only data that is given in view is accessible to user. So all data of base tables is not accessible to user which will give security of information. For example, [illegible] and view providing [illegible] in form of students.

(b) Hides complexity : The view may be made of any complex query. Hence instead of writing such complicated query again and again we can store such result as a view and [illegible] can hide the complexity of original query.

(c) Dynamic nature : View definition remains unaffected although there is any change in structure of a table. The dynamic nature does not hold true in case if base table is dropped or the column altered by view is altered.

Example : If view is made on two tables, selecting two columns from first table and two columns from second table. If we add one more column to first table then not cause any change on view.

(d) Does not allows direct access to the tables of data dictionary : This act like functionality of safeguard to data stored in the data dictionary. By this way user cannot change data dictionary in [illegible] database. Views can adapt to make data in data dictionary easily comprehensible and helpful.

(e) Data integrity : If data is accessed through a view the DBMS can automatically check the data to check for specified integrity constraints.

Q. 5 What are Disadvantages of Views ?

Ans. : Disadvantages of Views

(a) Performance : DBMS translates queries of view to queries on base table. Sometimes a simple query may take longer time to run. If view is defined by complex multi table query. As the complexity of query is hidden by view, users are not aware of how much complicated task the query is actually performing.

(b) View management : The view should be created as per standard then it will be simplifies the job of DBA. This happens generally when views are reference other views. Keep all information of all views in such case as at it it will become very difficult to manage views.

(c) Update restrictions : When a user tries to update a view, the DBMS must translate the query into an update on rows of the underlying base tables. Update is possible for simple views. Complex views cannot be updated as they are read-only type of views.

Q. 6 What are triggers ? Explain with example.

Dec. 2014, May 2017

Ans. : Triggers : A trigger is a procedure that is automatically invoked by the DBMS in response to specific alteration to the database or a table in database. Triggers are stored in database as a simple database objects. A database that has a set of associated triggers is called an active database. A database trigger enables DBA (Database Administrators) to create additional relationships between separate databases.

Components of Trigger (E-C-A model)

Event (E) : SQL statement that causes the trigger to fire (or activate). This event may be insert, update or delete operation database table.

Condition (C) : A condition that must be satisfied for execution of trigger.

Action (A) : This is code or statement that execute when triggering condition is satisfied and trigger is activated on database table.

Trigger syntax

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
[ENABLE | DISABLE]
<BEFORE | AFTER>
<INSERT | UPDATE | DELETE>
ON <table_name>
   [FOR EACH ROW]
DECLARE
   <variable_definitions>
BEGIN
   <trigger_code>
END;
```

OR REPLACE	If trigger is already present then drop and recreate the trigger
<trigger_name>	Name of trigger to be created.
BEFORE	Indicates that trigger is to be fired before the triggering event occurs.
AFTER	Indicates that trigger is to be fired after the triggering event occurs.
INSERT	Indicates that trigger is to be fired whenever insert statement adds a row to table.
UPDATE	Indicates that trigger is to be fired whenever Update statement modifies a row in a table.
DELETE	Indicates that trigger is to be fired whenever delete statement removes a row from table.
FOR EACH ROW	Trigger will be fired only once for each row.
WHEN	Contains condition that must be satisfied to execute trigger.
<trigger_code>	Code to be executed whenever triggering event occurs.

Fig. 3.2 : Trigger parameters

Trigger types

(a) Row level triggers : A row level trigger is fired each time the table is affected by the triggering statement. For example, if an UPDATE statement changes multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement. If a triggering statement do not affects any row then a row trigger will not run only. FOR EACH ROW clause is written that means trigger is row level trigger.

(b) Statement level triggers : A statement level trigger is fired only once on behalf of the triggering statement, irrespective of the number of rows in the table that are affected by the triggering statement. This trigger execute once even if no rows are affected.

For example, if a DELETE statement deletes several rows from a table, a statement-level DELETE trigger is fired only one time. This is Default type, when FOR EACH ROW clause is not written in trigger that means trigger is statement level trigger

Trigger example

Creating a trigger on employee table whenever new employee added a comment is written to EmpLog Table.

Example

```
SQL> CREATE OR REPLACE TRIGGER AutoRecruit
 2   AFTER INSERT ON EMP
 3   FOR EACH ROW
 4   BEGIN
 5   Insert into EmpLog values ('Employee Inserted');
 6   END;
 7   /

Trigger created.

SQL> INSERT INTO EMP
 2  VALUES (1,'Mahesh','Manager','1-JAN-
1986',3000,null,10);

1 row created.

SQL> SELECT * FROM EmpLog;
STATUS
------------------------
Employee Inserted
```

Consider other example, whenever there comes a new student add him to CS (Computer Science).

Example

```
SQL>CREATE TRIGGER CSAutoRecruit
  AFTER INSERT ON Student
  FOR EACH ROW
  BEGIN
  INSERT INTO Take VALUES (111, 'CS');
  END;
```

Trigger operations

(a) Data dictionary for triggers

Once triggers are created their definitions can be viewed by selecting it from system tables as shown below :

Syntax

```
MySQL>  Select   *
  From  User_Triggers
  Where Trigger_Name = '<Trigger_Name>';
```

This statement will give all properties of trigger including trigger code as well.

(b) Dropping Triggers

To remove trigger from database use command DROP

Syntax

```
MySQL>  Drop trigger <Trigger_Name>;
```

(c) Disabling Triggers

To deactivate trigger temporarily this can be activated again by enabling it.

Syntax

```
MySQL> Alter  trigger  <Trigger_Name>  {disable |
enable};
```

Trigger advantages

Triggers are useful for enforcing referential integrity, which preserves the defined relationships between tables when add, update, or delete the rows in those tables. Make sure that a column is filled with default information. After finding that the new information is inconsistent with the database, raise an error that will cause the entire transaction to roll back.

Trigger disadvantages

A trigger **hampers the performance of system** as database operations will go on slower due to triggering action.

Restrictions on triggers

Cannot modify the same table on which triggering event is written. Cannot modify a table which is connected to the triggering table by primary key foreign key relation.

Mutating table errors

This happens when the trigger is querying or modifying table whose modification activated the trigger, or a table that might need to be updated because of a foreign key constraint with a CASCADE policy. This problem is called as **mutating table problem.**

Error : ORA-04091: table name is mutating, trigger/function may not see it.

Q. 7 Explain security in SQL. **Dec. 2012**

Ans. : Security in SQL

A DBMS system always has a separate system for security which is responsible for protecting database against accidental or intentional loss, destruction or misuse.

Threats to Database

(a) **Confidentiality :** Data in database should be given to only authorized users.

Example : In HR department employee's personnel data should be accessible to that particular employee and the HR person only.

(b) **Integrity :** Only authorized users should be allowed to modify data.

Example : Only account department can change financial details of company.

(c) **Availability :** Authorized users can be able to accesses data any time he wants.

Example : Employee should be able to accesses own salary any time.

Security levels

(a) **Database level :** DBMS system should ensure that the authorization restriction needs to be there on users.

(b) **Operating system level :** Operating system should not allow unauthorized users to enter in system.

(c) **Network level :** Database is at some remote place and it is accessed by users through the network so security is required.

Security Mechanisms

(a) **Access control :** Which identifies valid users who may have any access to the valid data in the Database and which may restrict the operations that the user may perform e.g. ROLE function in SQL.

Example : The movie database might designate two roles : "users" (query the data only) and "designers" (add new data) user must be assigned to a role to have the access privileges given to that role.

Access privileges are assigned to users and roles. Each application is associated with a specified role. Each role has a list of authorized users who may execute / Design /administers the application.

(b) **Authenticate the user :** Which identify valid users who may have any access to the data in the Database. Restrict each user's view of the data in the database. This may be done with help of concept of views in relational databases.

(c) **Cryptographic control / Data encryption :** Encode data in a cryptic form (coded) so that although data is captured by unintentional user still he can't be able to decode the data. Used for sensitive data, usually when transmitted over communications links but also may be used to prevent by passing the system to gain access to the data.

(d) **Inference control :** Ensure that confidential information can't be retrieved even by deduction. Prevent disclosure of data through statistical summaries of confidential data.

(e) **Flow control or physical protection :** Prevents the copying of information by unauthorized person. Computer systems must be physically secured against any unauthorized entry.

(f) **Virus control :** At user level authorization should be done to avoid intruder attacks through humans. There should be mechanism for providing protection against data virus.

(g) **User defined control :** Define additional constraints or limitations on the use of database. These allow developers or programmers to incorporate their own security procedures in addition to above security mechanism.

Q. 8 Explain Authorization in SQL.

May 2012, Dec. 2012

Ans. : Authorization in SQL

Authorization is finding out if the person, once identified, is permitted to have the resource. Authorization explains that what can do and is handled through the DBMS unless external security procedures are available. This is usually determined by finding out if that person is a part of a particular group, if that person has paid admission, or has a particular level of security clearance. Authorization is equivalent to checking the guest list at an exclusive party, or checking for ticket when go to the opera. Database management system allows DBA to give different access rights to the users as per their requirements. In SQL Authorization can be done by using read, insert, update or delete privileges.

Basic authorizations

Use any one form or combination of the following basic forms of authorizations,

(a) **Resource authorization :** Authorization to access any system resource. **Example :** Sharing of database, printer etc.

(b) **Alternation authorization :** Authorization to add attributes or delete attributes from relations.

(c) **Drop authorization :** Authorization to drop a relation.

Mandatory access control

This access control mechanisms can be explained as objects (e.g. tables, views, and rows), subjects (e.g. users, programs), security classes and clearances. Each database object is assigned a security class, which define its security parameters and each subject is assigned clearance for a security class; Comparing discretionary access control and mandatory access control

ROLE Based Access Control (RBAC)

In Role-Based Access Control (RBAC), access decisions are based on an individual's roles and responsibilities within the organization or user base. The process of defining roles is usually based on analysing the fundamental goals and structure of an organization and is usually linked to the security policy.

For instance, in an educational organization, the different roles of users may include those such as staff, faculty, attendant, student, principal, etc. Obviously, these members require different levels of access in order to perform their functions, but also the types of web transactions and their allowed context vary greatly depending on the security policy and any relevant.

Discretionary Access Control

To maintain access rights of each database user discretionary access control is used. It allows multiple users to access some object (i.e. grants the privileges) in a specified mode, like read, write or combination of these. Privilege is set of actions that a user can perform on a database object are called the privileges. It is right to execute particular SQL statement on database. The high level user (Like DBA) has power to grant access to database and its object. This access control is done by two commands GRANT and REVOKE. GRANT command helps us to give user privileges to base table and views.

Account creation :

Allows to create a new user account and password for same. Allows to create a new user group (Role).

Grant Privilege : DBA can grant some privilege to certain user or role.

Revoke Privilege : DBA can revoke (cancel) all privileges from user or role which are given by him.

Q. 9 Explain the detailed concept of Database connectivity using JDBC.

Ans. :

JDBC (Java Database Connectivity)

(a) **Java Database Connectivity (JDBC)** is a java API enables the java programs to execute SQL statements.

(b) JDBC provides some methods for querying and updating the data in Database Management system such as SQL, Oracle etc.

(c) JDBC connections which supports creating and executing statements such as SQL INSERT, UPDATE and DELETE.

(d) Driver Manager is the backbone of the JDBC architecture.

Types of JDBC Drivers

JDBC Type 1 : JDBC-ODBC Bridge Driver

1. **Architecture :** Lower level JDBC Driver which makes use of he ODBC driver to connect to the database. The driver converts JDBC method calls into ODBC calls and sends them to the ODBC driver. This implies that the ODBC driver, as well as the client database code, installed on the client machine.

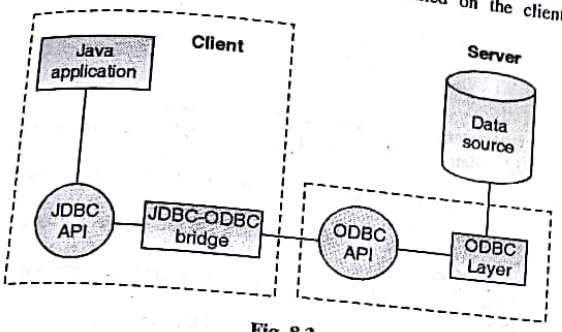


Fig. 8.3

2. **Advantages :** Many data base contains ODBC drivers so can be implemented generally. Easy Installation Very useful for which only a small number of clients access to the database.
3. **Disadvantages :** Partially written in Java which makes it not portable. Not suitable for Web Applications. Performance overhead as JDBC calls is converted into ODBC calls and then passed on to the ODBC driver. ODBC driver required on the client machine

JDBC Type 2 : Native-API/Partly Java Driver

1. **Architecture :** Lower-level JDBC Driver API that makes use of the client-side libraries of the database to connect. The driver converts JDBC calls into database-specific calls. The type 2 driver is not written entirely in Java as it interfaces with non-Java code that makes the final database calls.

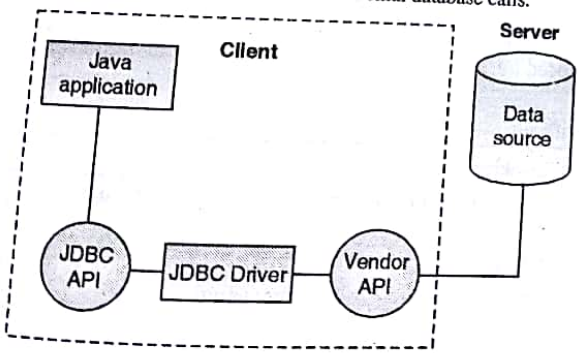


Fig. 8.4

2. **Advantages :** No ODBC drivers are required.
3. **Disadvantages**
   i. Database vendor specific client API libraries must be installed on the client machine.
   ii. Not all the database vendors provide client API libraries.
   iii. Not suitable for Web Applications.

JDBC Type 3 : Network Protocol Driver

Known as the Pure Java Driver for database Middleware.

Three-tiered approach where the client code sends the JDBC calls through the network to a middle-tier server. The middle-tier converts the JDBC calls directly or indirectly into database specific protocol, which will forward the JDBC calls to the database.

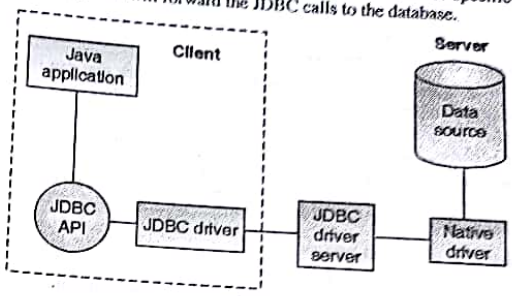


Fig. 8.5

1. **Advantages**
   i. No database library is required on the client side.
   ii. It can provide additional services like caching, auditing etc.
   iii. Well suited for web applications
2. **Disadvantages**

   The additional layer might be a bottleneck for the JDBC calls.

Type 4 : Pure Java Driver

Completely implemented in java. Driver converts JDBC calls into vendor specific DBMS protocol so that client applications can directly communicate with database server. It is also called as thin client driver as all processing at server. At client side no processing related to database occurs.

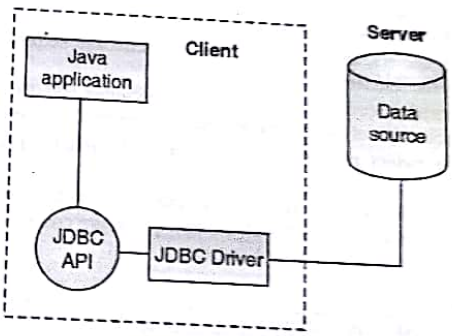


Fig. 8.6

1. **Advantages**
   i. Platform independent
   ii. Eliminate deployment administration issues.
   iii. Most suitable for web application.
   iv. Very good performance
   v. No need to install special software on client machine.
2. **Disadvantages**
   i. User needs a different driver for each database.

ii. May not be suitable for some applications if the underlying protocol does not handle issues such as security and network connectivity well.

Java Database Connectivity Steps

1. **Define the Connection URL**
2. Establish (open) database the Connection
3. Create a Statement Object
4. Execute (SQL) Query or Update
5. Display the query results
6. Close the Connection

Chapter 9 : PLSQL

Q. 1 Write short note on stored procedure.

Ans. : Stored Procedure

A stored procedure is a named PL/SQL block that can take some parameters (arguments) and produce some output once it is called using stored procedure execution. Stored procedures are just text objects, and don't store any data. Procedures provides access to data and returns datasets, just like a view. Generally speaking you use a procedure to perform a specific action. The stored procedure can call data from Table or view.

Benefits of stored procedures

1. Stored procedure offers modularity of code.
2. Procedures promote better reusability and maintainability for code.
3. Once validated, they can be used in any number of times without compiling again and again in order to make execution faster.
4. It can be used in any number of applications to make faster data access.
5. If the definition changes only the procedure code is affected. So it will be simple for maintenance.

Q. 2 Write short note on implementation of stored procedure.

Ans. : Creating Stored Procedure

1. A stored procedure has a header, a declaration section, an executable section, and an optional exception-handling section.
2. We can create new procedures with help of the CREATE PROCEDURE statement, which may declare a list of parameters, and also must define the actions to be performed by the standard PL/SQL block.
3. Stored procedure blocks can start with either BEGIN or the declaration of local variables and end with END statment.
4. We cannot reference any host or bind variables inside a stored procedure.
5. Using REPLACE option if that procedure exists, it will be dropped and replaced with the new procedure created by the statement.

Syntax

```
CREATE [OR REPLACE] PROCEDURE procedure_name
 [(parameter1 [mode1] datatype1, parameter2 [mode2]
datatype2)]
AS
BEGIN
   PLSQL CODE;
END;
```

Example : Create a procedure to increment salary of all employees by 10%.

```
SQL>CREATE PROCEDURE spIncrSalary
   AS
   Begin
     UPDATE  Emp
     SET     Sal = 1.1*Sal
     WHERE   Job = 'MANAGER';
   End;
```

6. How to create a stored procedure using oracle
   a. Enter the text of the CREATE PROCEDURE statement and save it as a script file (*.sql* extension).
   b. From SQL*Plus, run the script file to compile the source code.
   c. Use SHOW ERRORS command to see any compilation errors.
7. How to create a stored procedure using oracle SQL command line
   a. Connect to oracle server by providing valid credantials.
   b. Write a valid code for stored procedure using above syntax.
   c. Use SHOW ERRORS command to see any compilation errors.

Executing Stored Procedure

1. We can execute the stored procedure with help of EXEC command or with help of any application program.
2. Executing stored procedure with Oracle

```
SQL>    EXEC spIncrSalary;
```

3. Invoking a procedure from PL/SQL Block or from other procedure.

```
DECLARE
. . . .
BEGIN
   spIncrSalary;                    -- Invoke procedure from
procedure

END;
```

4. Invoking a procedure from another procedure

```
SQL>CREATE OR REPLACE PROCEDURE access_stud
   IS
   BEGIN
     spIncrSalary;  -- Invoke procedure

   END;
```

Q. 3 Explain how to pass parameters of stored procedures

Ans. : Parameter Types in Stored Procedure

1. **Formal parameters :** Variables declared in the parameter list of a stored procedure are called as Formal parameters.

Example

```
SQL> CREATE PROCEDURE raise_Fees (p_id NUMBER,
p_fees NUMBER)
  Begin
    ...
  END;
```

2. **Actual parameters :** Variables or expressions referenced in the parameter list of a stored procedure call are nothing but actual parameters.

Example

```
SQL>  EXEC raise_fees ('1001', 2000);
```

Q. 4 Explain how we can change definition of stored procedures.

Ans. : Altering Stored Procedure

a. We can overwrite definition of stored procedure by using ALTER command.
b. It also changes procedure details in data dictionary.
c. We can change type of parameter in alter option.
d. If existence of procedure in not known then we will use option CREATE OR REPLACE which will create new procedure if not existing otherwise replace original procedure.

Syntax

```
CREATE OR REPLACE PROCEDURE procedure_name
[(parameter1 [mode1] datatype1, parameter2 [mode2]
datatype2)]
AS
BEGIN
  PL/SQL Statements;
END;
```

Example

Write a procedure to increase 20% fees of student having given id number.

```
SQL>CREATE OR REPLACE PROCEDURE Raise_Fees
  (v_id IN stud.studno%TYPE)
  IS
  BEGIN
          UPDATE  stud
          SET fees = fees * 1.20
          WHERE   studno = v_id;
  END;
```

Q. 5 Write difference between stored procedure and Function.

Ans. : Compare stored procedure and Function

Sr. No.	Stored Procedure	Stored Function
1	Invoke as a PLSQL Block of code or using execute statement	Executed as a part of PLSQL or SQL expression.
2	May not return any parameter	Always return parameter
3	It can return one or more output as variables	It must return atleast one variable.
4	It can contain one or more RETURN statements.	It must contain a single RETURN statement.

Q. 6 What is PLSQL Cursors ? Explain various types of cursors with example.

Ans. : Cursor

A **cursor** is a method by which we can assign a name to a result of SELECT statement and manipulate the data within that SQL statement. Oracle server uses SQL work areas or result set to execute SQL statements and store its processing information. A PL/SQL cursor allows to name a work area or result set and accesses its stored information.

There are two main types of cursors,

Implicit Cursor

Every SQL DML (Data Manipulation Language) and DRL (Data Retrieval Language - SELECT) statement executed by the Oracle server is having an individual cursor associated with it. PL/SQL implicitly declares a cursor for all SQL SELECT and DML statements, including queries that return only one row. An implicit cursor is opened and closed by the oracle server automatically to process each SQL statement. For example whenever execute any SELECT statement or any of INSERT, DELETE or UPDATE statement oracle server automatically declares on cursor associate with it.

Explicit Cursor

Use explicit cursors to process each row individually returned by a multiple-row SELECT statement in cursor definition. Explicit cursor allows us to name a work area and access its stored information. Explicit cursors are used in query which returns many rows. The set of rows in table fetched by a cursor query is also called active set. The size of the active set is rows returned by a select statement. Explicit cursor is declared in the DECLARE section of PL/SQL block.

Q. 7 Explain Implicit cursor attributes in details with example.

Ans. : Implicit Cursor Attributes

An Oracle server implicitly opens and closes such cursors to process each SQL statement. Every SQL implicit cursor has many attributes which returns some useful information about the execution of a data manipulation statement or select statement.

Sr. No.	Attribute
1.	SQL%ISOPEN
2.	SQL%FOUND
3.	SQL%NOTFOUND
4.	SQL%ROWCOUNT

SQL%ISOPEN

This attribute will return Boolean output i.e. TRUE or FALSE. This statement will always returns FALSE, as the database closes the SQL cursor automatically after executing its associated SQL statement.

SQL%FOUND

This attribute will return Boolean output i.e. TRUE or FALSE. This statement will returns TRUE if an INSERT, UPDATE, or DELETE statement is affecting more then one row or a SELECT INTO statement returned more then one row. Otherwise, it returns FALSE.

SQL%NOTFOUND

This attribute will return Boolean output i.e. TRUE or FALSE. This statement will returns FALSE if an INSERT, UPDATE, or DELETE statement is affecting more then one row or a SELECT INTO statement returned more then one row. Otherwise, it returns TRUE.

SQL% ROWCOUNT

This attribute will return integer output. This statement will exactly return the number of rows affected by an DML statement, or returned by a SELECT INTO statement.

Example

Write a PLSQL block to increment salary of all employees above age 25 in company by 5000 and print number of rows updated by above DML operation.

```
-- Option to Print Output
SQL> SET SERVEROUTPUT ON;

-- Optional declarative section
SQL> DECLARE
 V_number NUMBER(6);
-- Executable section
BEGIN
   UPDATE Emp
   SET Sal = Sal + 5000
   WHERE age > 25;
   V_number:= SQL%ROWCOUNT;
 IF SQL%FOUND THEN
  DBMS_OUTPUT.PUT_LINE('Salary of ' ||v_number || '
 Employees updated.');
 ELSE
  DBMS_OUTPUT.PUT_LINE('No Employees having age
 above 25');
 END IF;
 END;

 SQL> /
 Salary of 20 Employees updated.
 PL/SQL procedure successfully completed.
 successfully completed.
```

Q. 8 Explain cursor for loops in details with example.

Ans. : Cursor FOR Loops

FOR loop will have a control statement before LOOP keyword which determine the number of rows fetched by a cursor. In case of FOR loop there is no need to declare the record variable in code; it is declared implicitly as a cursor_name%ROWTYPE by system itself.

Refer the record variable inside loop only; and use an expression to reference the existing value of this record set variable.

Syntax

```
FOR record_name IN cursor_name
LOOP                          // Start of Loop
   Statement 1;    // Statement
....
END LOOP;                     // End of Loop
```

FOR Loop is shortcut for above type of loops as there is no need to OPEN, CLOSE and FETCH cursor explicitly it will be done automatically. Even Record set variable is declared automatically.

No need to write terminating condition as soon as cursor reaches to last row it will exit from fetch operations and closes cursor automatically.

Example : A PLSQL block to print employee name and salary of all employees above given age.

```
-- Option to Print Output
SQL> SET SERVEROUTPUT ON;

SQL> DECLARE
  CURSOR Emp_Cur
  IS SELECT Ename,Sal
    FROM  EMP
    WHERE  age > &Age;
 BEGIN
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  DBMS_OUTPUT.PUT_LINE('Name      | salary');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  FOR Emp_Rec IN Emp_Cur
  LOOP
  DBMS_OUTPUT.PUT_LINE(RPAD(Emp_Rec.ename,10,' ')||'|
'||Emp_Rec.sal);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('-------------------------');
  DBMS_OUTPUT.PUT_LINE(Emp_Cur%ROWCOUNT||'     rows
Selected.');
  DBMS_OUTPUT.PUT_LINE('-------------------------');
 END;

 SQL> /
 Enter value for age: 25
 ------------------------
 Name      | Salary
 ------------------------
 SMITH     | 10800
 ALLEN     | 11600
 WARD      | 11250
 JONES     | 12975
 MARTIN    | 11250
 BLAKE     | 12850
 CLARK     | 12450
 ------------------------
 7 rows Selected.
 ------------------------
 PL/SQL procedure successfully completed.
```

Cursor FOR loops with subquery : In FOR loop declare subquery in place of cursor variable. So any query that is solved by cursor can be solved without use of cursor by using cursor query in FOR loop.

Syntax

```
FOR record_name IN (SubQuery)
LOOP          -- Start of Loop
Statement 1;  -- Statement
....
END LOOP;     -- End of Loop
```

FOR Loop is shortcut for above type of loops as there is no need to OPEN, CLOSE and FETCH cursor explicitly it will be done automatically. Even Record set variable is declared automatically. No need to write terminating condition as soon as subquery reaches to last row it will exit from fetch operations.

Example : PLSQL block to print employee name and salary of all employees above given age.

```
-- Option to Print Output
SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
 DBMS_OUTPUT.PUT_LINE('-------------------------');
 DBMS_OUTPUT.PUT_LINE('Name      | salary');
 DBMS_OUTPUT.PUT_LINE('-------------------------');
 FOR Emp_Rec IN (SELECT Ename,Sal FROM EMP WHERE
age>&Age)
 LOOP
 DBMS_OUTPUT.PUT_LINE(RPAD(Emp_Rec.ename,10,' ')||'|
'||Emp_Rec.sal);
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('-------------------------');
END;
Output
SQL> /
Enter value for age: 25
-------------------------
Name        | salary
-------------------------
SMITH       | 10800
ALLEN       | 11600
WARD        | 11250
JONES       | 12975
MARTIN      | 11250
BLAKE       | 12850
CLARK       | 12450
-------------------------
PL/SQL procedure successfully completed.
```

Q. 9 Explain cursor variables in details.

Ans. : Cursor Variables : Cursor variables are like C pointers, which hold only memory location (address) instead of the item itself. So, declaring a cursor variable creates only a pointer and not an item.

In PL/SQL, a pointer has datatype as,

REF X

where REF = REFERENCE
X = class of objects.

Hence, a cursor variable has datatype called as a REF CURSOR. An explicit cursor which names the work area or we can also use a cursor variable which points to the work area.

A cursor always refers to the same work area. But a cursor variable can refer to different work areas. So both are operationally different. Cursor variables to pass query result sets between various PL/SQL stored subprograms and clients. So they only shears a pointer to the query work area in which the result set is stored.

A query work area accessible till the time cursor variable points to it. Therefore, pass the value of a cursor variable from one scope to another easily. Cursor variables are available for use to every client in database environment. Pass cursor variables from application software to the database server using remote procedure calls.

Syntax :

Define a REF CURSOR type
TYPE **ref_type_name** IS REF CURSOR
[RETURN return_type];
Declare a cursor variable of that type
ref_cv ref_type_name

Chapter 10 : Relational Database Design

Q. 1 Describe various design guidelines for relational schema.

Ans. : Design Guidelines for Relational Schema

(1) Goodness of Relational Design : In process of database design, to develop some measure of goodness for the quality of the design. The above measure will help the developer to analyze why one grouping of attributes is better than another grouping of attributes. There are two levels for measuring goodness of relation schemas.

Logical (or Conceptual) Level : The goodness of relational schema depends on how users understand the meaning of various attributes of relation. This will helps users to understand the meaning of the data stored in the relations, and hence it will make easy to formulate relational queries correctly.

Implementation (or Physical Storage) Level : It helps user to understand how the tuples in a base relation are stored and updated. This level is applied to base relational schema stored as files.

(2) Database Design Approach : The database design can be done using the bottom-up approach to design relational schema using individual attributes of relation or top-down approach to identify individual attributes from relational schema.

Bottom-up design methodology / Design by Synthesis : The basic relationships among individual attributes are identified then it will be combined to construct relation schemas. This is not very popular approach as it needs to study many binary relationships among attributes at the beginning which is very difficult.

Top-down design methodology / Design by Analysis : The starts with relational schema and it will be further decomposed to a group of attributes to achieve desirable properties. This is very practical approach and used in real world database projects.

(3) Guidelines for Relational Schema : To determine the quality of relation schema design some informal guidelines can be used.

Guideline 1 : Clear semantics of the attributes in relational schema : Semantics of attribute should be very clear in relational schema so that relational schema will have some real-world meaning associated with it. The relational schema has a clear meaning associated with it.

Example,

Employee (Emp_Id, Ename, Address, Salary)

The Employee table contains information about all employees in company with their address and Salary.

Guideline 2 : Reducing the Redundant Data in Tuples

A relational schema may have some redundancy in database design, if it stores data redundantly. If same data is stored at more then one locations will leads to redundancy and wastage of memory space.

Data Anamolies: An inconsistent data may cause some problems while adding, updating or deleting data in table which is called as data anomalies.

Redundant data is more vulnerable to various data anomalies as if data is updated at only one location and not at other locations, then that data becomes inconsistent, and this problem referred to as an update anomaly. A normalized database stores non-primary key data in only one location. A relational database table should avoid all data anomalies.

Example,

Employee (Emp_Id, Ename, Address)

Emp_Salary (Emp_Id, Ename, payScale, grossSalary, netSalary)

Emp_Designation (Emp_Id, Ename, Desg, fromDate, toDate)

(a) Update Anomaly : The relational schema may have same data stored in multiple relations, if we update such data from only one relation may result in logical inconsistencies.

In above example,

All 3 tables contain the Ename attribute, thus any change in name of one employee will lead to updating his name in all 3 tables. Otherwise, if all the records are not updated then some tables may leave in an inconsistent state.

(b) Insertion Anomaly : There is a possibility in which certain facts cannot be recorded in database. An Insert Anomaly arises when certain attributes cannot be added into the database without the presence of other attributes.

In above example,

It is not possible to add a row in Emp_Salary table or Emp_Designation table for an employee who is not exists in employee table.

(c) Deletion Anomaly : If data deleted from one table all relevant data in another related tables must also be deleted otherwise it will create data inconsistency problem.

Deletion of some data from one relation necessitates the deletion of some other data in other table.

In above example,

It is not possible to delete a row in Employee table if Emp_Salary table or Emp_Designation table contains data for respective employee.

Guideline 3 : Reducing Null values in Tuples :

A value of NULL is different from an empty, White spaces (blank spaces) or zero value. Null values in tuple will cause wastage of memory space and it will also create problem of understanding. Relations should be designed in such a way that their tuples should not contain any NULL values. We can at least try make number of NULL values as low as possible. Attributes with NULL values can be placed in separate relations with the primary key. There are certain resons for Null values,

Not applicable data

Invalid data

Unknown data or data not available

Guideline 4 : Dissallowing Spurious Tuples :

The bad designs of a relational database may result in erroneous results for some JOIN operation. As it is not possible to get original relation data from new relation after JOIN operation. The relational schema must be designed to satisfy the property of lossless join. If original relation contains fewer number of tuple then tuples generated by doing a natural-join of original relations.

Q. 2 Explain concept of functional dependency.

Ans. : Functional Dependencies

The concept of functional dependency is given by E. F. Codd. Functional Dependency (FD) provides a constraint between various attributes of a relation. Functional dependencies are restrictions imposed between two set of attributes in relation from a database. In a Relation R with attributes X and Y represented as R(X, Y), where Y is functionally dependent on other column X or we can say X functionally determines Y.

This dependency can be denoted with help of arrow (→)

$$X \rightarrow Y$$

The data value in column Y must change when data value in another column X is modified. All the attributes before arrow is called as **determinant** and attributes after arrow is called as determine.

Example

Consider an employee table with columns as shown in Table 10.1

Table 10.1 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

Case 1 : (X→Y)

Consider an Employee table for specific employee_Id there is one and only one Ename whereas for another employee_Id there can be other Ename.

$$\text{Employee\_Id} \rightarrow \text{Ename}$$

As per above constraint, it is possible to have multiple employees with same Ename and different Employee_Id. But it is not allowed to have two employees with same Employee_Id and different Ename.

Case 2 : (X→YZ)

In above Employee table using below given functional dependency, for specific employee_Id there is one and only one set

of Ename and Salary whereas for another employee_Id there can be other values of Ename and salary.

Employee_Id → Ename, Salary

As per above constraint, it is possible to have multiple employees with same Ename and Salary.

Case 3 : (XY→ZW)

In above Employee table using below given functional dependency, for one employee_Id and Project_Id pair there is only one amount of time spent (Hours) and allowance given by company whereas for another pair there can be other values of Hours and Allowance.

Employee_Id, Project_Id → Hours, Allowance

As per above constraint, it is possible to have multiple employee_Id and Project_Id pairs with same values of Hours and Allowance.

Q. 3 List all functional dependencies satisfied by the relation. Dec. 2013, May 2015

a	b	c
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

Ans. :

To find out all functional dependencies satisfied by the relation, we must remember one determinant (attributes before arrow) will give one and only one value of determine (attribute after arrow).

A	B	C	Tuple
a_1	b_1	c_1	Tuple 1
a_1	b_1	c_2	Tuple 2
a_2	b_1	c_1	Tuple 3
a_2	b_1	c_3	Tuple 4

Step 1 : Test for type 1 FD (X→Y)

Functional Dependency	Violated First by Tuple	Result
A → B	--	Valid FD
A → C	Tuple 2 $a_1 \rightarrow c_1$ $a_1 \rightarrow c_2$	Not Valid FD
B → A	Tuple 3	Not Valid FD
B → C	Tuple 2	Not Valid FD
C → A	Tuple 3	Not Valid FD
C → B	--	Valid FD

Step 2 : Test for type 1 FD (X→YZ)

Functional Dependency	Violated First by Tuple	Result
A → BC	Tuple 2 $a_1 \rightarrow b_1c_1$ $a_1 \rightarrow b_1c_2$	Not Valid FD
B → AC	Tuple 2	Not Valid FD
C → AB	Tuple 3	Not Valid FD

Step 3 : Test for type 1 FD (XY→Y)

Functional Dependency	Violated First by Tuple	Result
AB → C	Tuple 2 $a_1b_1 \rightarrow c_1$ $a_1b_1 \rightarrow c_2$	Not Valid FD
AC → B	--	Valid FD
BC → A	Tuple 3	Not Valid FD

Therfore, all FDs satisfied by relation are,

A → B; C → B; AC → B

Q. 4 Consider the following relation. May 2014

A	B	C	Tuple #
10	b_1	c_1	#1
10	b_2	c_2	#2
11	b_4	c_1	#3
12	b_3	c_4	#4
13	b_1	c_1	#5
14	b_3	c_4	#6

Given the previous state which of the following dependencies may hold in the above relation ? If the dependency cannot hold explain why by specifying the tuples that cause the violation :

(i) A →B (ii) B →C (iii) C → B

(iv) B → A (v) C → A

Ans. :

To find out all functional dependencies satisfied by the relation, we must remember one determinant (attributes before arrow) will give one and only one value of determine (attribute after arrow).

(i) A →B

As in above relation to test A → B, {10} → {b1} but in Tuple # 2 {10} → {b2}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #2.

(ii) B → C

As in above relation to test B → C, {b1}→ {c1}, {b2}→{c2}, {b3}→{c4} for all tuples.

Therefore, Dependency holds for all tuples in above table.

(iii) C → B

As in above relation to test A → B, {c1} → {b1} but in Tuple #3 {c1} → {b4}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #3.

(iv) B → A

As in above relation to test A → B, {b1} → {10} but in Tuple #5 {b1} → {13}

Also {b3} → {13} but in Tuple #6 {b3} → {14}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #5 and Tuple #6.

(v) C→ A

As in above relation to test C→ A, In Tuple #1 {C1} → {10,11,13}

But in Tuple #3 {c1} → {11} and in Tuple #5 {c1} → {13}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #3 and Tuple #5.

Q. 5 What are types of functional dependencies ?

Ans. : Types of Functional Dependencies

(1) Full functional dependency : A functional dependency is a full functional dependency if removal of any attributes from determinant will invalidate the dependency. A functional dependency A → B is a full functional dependency if removal of any attributes from A means that the dependency does not hold any more.

Example : Consider an employee table with columns as shown in Table 10.2.

Table 10.2 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

In the above example, Hours and allowance are fully functionally dependent on both Employee_Id and Project_Id.

Employee_Id, Project_Id → Hours, Allowance

The number of hours spent on the project by a particular employee can not be determined with the project number (Project_no) alone. It needs the employee number (Emp_no) as well.

(2) Partial functional dependency : A partial dependency means that a non key column is depend on some columns in composite primary key of a table. An FD A → B is a partial dependency if there is some attribute X ∈ A (X subset of A), that can be removed from A and the dependency will still hold. If determine (attributes after arrow) attributes depends on **part of** (partial) determinant attributes. Such dependency is called as partial functional dependency.

Example : Consider an employee table with columns as shown in Table 10.3.

Table 10.3 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

In the above example, attribute salary is considered to be functionally dependent on both Employee_Id and Project_Id.

Employee_Id, Project_Id → Salary

But, Attribute salary functionally dependent on Employee_Id is also holds true.

Employee_Id → Salary

So, Salary is partial functionally dependent on attribute pair Employee_Id and Project_Id.

(3) Transitive dependency : If one attribute of relation is functionally dependent on other dependent attribute then such a dependency is called as **transitive dependency**. An FD X →Y in a relation R is a transitive dependency, if there is a set of attributes Z that is not a subset of any key of R, and both X → Z and Z → Y holds true.

Example : Consider an employee table schema,

Employee_Id → Department_Id

Department_Id → Dname

Table 10.4 : Employee Table

Employee_Id	Ename	Salary	Department_Id	Dname
10	Mahesh	50000	C1	IT
12	Suresh	25000	E2	HR
15	Ganesh	26000	C1	IT
18	Mahesh	50000	E2	HR

Dependency of Department_Id on key attribute Dname is transitive functional dependency as Dname is depends on Department_Id which is depend on Employee_Id. So, Dname is transitive functionally dependent on Employee_Id.

(3) Trivial functional dependency : Functional dependency (FD) X → Y and Y is a subset of X, then it is called as a trivial FD. Functional dependency X → Y and Y is not a subset of X, then it is called as a non-trivial functional dependency.

Example : For employee table the below given FD is trivial as Ename is a subset of {Employee_Id, Ename}

Employee_Id, Ename → Ename

And below given FD is non-trivial as Hours is not a subset of {Employee_Id, Project_Id}

Employee_Id, Project_Id → Hours

(4) Multivalued dependency : Multivalued dependency defined by X →→ Y is said to hold for a relation R (X, Y, Z) if for a given set of values of X, there is a set of associated values of attribute Y, and X values depend only on X values and have no dependence on the set of attributes Z. **Multivalued dependency** in turn, is defined as relationship which accepts the cross product pattern.

Example : For employees car table as given below,

Table 10.5 : Employee_Car Table

Employee_Id	Ename	Car
10	Mahesh	Ertiga
12	Suresh	Zen
15	Ganesh	Sentro
10	Mahesh	Wagon R

The FDs are given as below,

Employee_Id →→ Car

α →→ β says relationship between α and β independent of relationship between α and R-β. That means **Employee_Id →→ Car** relationship is independent of Employee_Id and EName relation. i.e. **Employee_Id →→ Car** is independent of Employee_Id →→ Ename

Q. 6 Give Armstrong axioms.

OR List the Armstrong's axioms for functional dependencies. What do you understand by soundness and completeness of these axioms ?

Ans. : FD Properties (Armstrong's Axioms / Closures of FD)

Given that Relation R(X, Y, Z, W); represents a table R with set of indivisible attributes X, Y, Z and W. It is possible to derive many properties of functional dependencies. Axioms are nothing but rules of inference which provides a simple technique for reasoning about functional dependencies.

(1) Primary Properties

a. Subset property (Axiom of Reflexivity)
For given relation R(X, Y, Z, W),
If Y is a subset of X as shown in diagram,

Then $X \rightarrow Y$
(Which can be referred as X is functionally dependent on Y)

b. Append Property (Axiom of Augmentation)
For given relation R(X, Y, Z, W),
If $X \rightarrow Y$
Then $XZ \rightarrow YZ$
It is possible to append attribute Z to both sides of FD provided that it is part of same table.

c. Transitivity (axiom of transitivity)
For given relation R(X, Y, Z, W),
If $X \rightarrow Y$ and $Y \rightarrow Z$
Then $X \rightarrow Z$
It is possible to use transitivity if attribute X, Y and Z are part of the same table.

(2) Secondary Properties

a. Union
For given relation R(X, Y, Z, W),
If $X \rightarrow Y$ and $X \rightarrow Z$
Then $X \rightarrow YZ$

b. Decomposition
For given relation R(X, Y, Z, W),
If $X \rightarrow YZ$
Then $X \rightarrow Y$ and $X \rightarrow Z$

c. Pseudo Transitivity
For given relation R(X, Y, Z, W),
If $X \rightarrow Y$ and $YZ \rightarrow W$
Then $XZ \rightarrow W$

Q. 9 What is decomposition ?

Ans. : Decomposition

(a) If a relation is not in the normal form and we wish the relation to be normalised so that some of the anomalies can be eliminated, it is necessary to decompose the relation in two or more relations.

(b) This is a process of dividing one table into multiple tables can be done using projection operator.

(b) Decomposed table can be reconstructed using join operation.

Q. 10 What is Normalization ? May 2016

Ans. : Normalization Process

Normalization is a step by step decomposition of complex records into simple records. Normalization is a process of organizing data in database in more efficient form. It results in tables that satisfy some constraints and are represented in a simple manner. This process is also called as canonical synthesis. Normalization is a step by step decomposition and database designers may not normalize relation to the highest possible normal form. The relations may be left in a lower normal form like 2NF, which may cause some penalties like data anomalies.

Definition

Normalization is a process of designing a consistent database by minimizing redundancy and ensuring data integrity through decomposition which is lossless.

Goals of Database Normalization

1. Ensures data integrity.
2. Prevents redundancy in data.
3. To avoid data anomaly
   (a) Update anomaly
   (b) An insertion anomaly
   (c) Deletion anomaly

Q. 11 Explain 1NF. May 2016

Ans. :

First Normal Form (1NF)

This is simplest form of normalization, simplifies each attribute in relation. This normal form given by E.F. Codd (1970) and the later version by C.J. Date (2003).

Definition

A relation is in 1NF, if every row contains exactly one value for each attribute. 1NF states that all attributes in relation must have atomic (indivisible) values and all attribute in a tuple must have a single value from the domain of that attribute. In short rules for data in 1NF is,

A columns in a table should contain only indivisible data.

Example

Consider an employee table with columns as shown in diagram, The relational schema not in 1 NF is represented as,

Employee Table

Employee_Id	Ename	Salary	Ecity

The state of Employee relational schema is as given below and it contains the Ecity which is non atomic (divisible) domain.

Table 10.6 : Non-Normalised Employee Table

Employee_Id	Ename	Salary	Ecity
10	Mahesh	50000	Mumbai, Pune
12	Suresh	25000	Mumbai
15	Ganesh	26000	Pune
18	Kasturi	50000	Mumbai, Delhi

To convert relational schema in 1NF, the Ecity attribute is divided in atomic domains it may introduce some data redundancy.

Fig. 10.7 : 1NF Employee Table

Employee_Id	Ename	Salary	Ecity
10	Mahesh	50000	Mumbai
10	Mahesh	50000	Pune
12	Suresh	25000	Mumbai
15	Ganesh	26000	Pune
18	Kasturi	50000	Mumbai
18	Kasturi	50000	Delhi

Minimizing Domain Redundancy

The first normal for will solve the group redundancy occurs in domain value as it allows only a single value from the domain of that attribute. So, 1NF will solve all problems related to domain redundancy. Nested relations must be removed to convert relation in 1 NF.

Q. 12 Explain 2NF. May 2016

Ans. : Second normal Form(2NF)

This normal form makes use of full functional dependency and tries to remove problem of redundant data introduced by 1NF decomposition. This normal form is given by E.F. Codd in 1971.

Definition

A relation is in 2NF, if it is in 1NF and all non-key attributes in relation are fully functionally dependent on the primary key of the relation.

OR

A relation is in 2NF, if it is in 1NF and every non-key attribute is fully functionally dependent on the complete primary key of relation (and not depends on part of (partial) primary key).

In short 2NF means,

It should be in 1NF.

There should not be any partial dependency on primary key attributes.

Example

Consider an employee table with columns as shown in diagram,

The relational schema **not in 2 NF** is represented as,

Consider an Employee table with following FDs,

Employee_Id → Ename, Salary

Employee_Id, Project_Id → Hours, Allowance

As

{Employee_Id, Project_Id} → Ename, Salary, Hours, Allowance

Therefore,

Candidate key {Employee_Id, Project_Id} is selected as primary key.

As attributes Hours, Allowance of employee table are full functionally dependent on primary key whereas attributes Ename and Salary are partially depends on primary key. (As Ename, Salary are depends **on part of primary key**)

The state of Employee relational schema is,

Table 10.8 : Non-2NF Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000
15	Ganesh	26000	E001	24	20000
18	Mahesh	50000	B056	11	10000

To normalize above schema to 2NF we can decompose tables as ,

Employee (Employee_Id, Ename, Salary)

Employee_Id → Ename, Salary

Table 10.9 : 2NF Employee Table

Employee_Id	Ename	Salary
10	Mahesh	50000
12	Suresh	25000
15	Ganesh	26000
18	Mahesh	50000

Project (Employee_Id, Project_Id, Hours, Allowance)

Employee_Id, Project_Id → Hours, Allowance

Table 10.10 : 2NF Project Table

Employee_Id	Project_Id	Hours	Allowance
10	E001	44	40000
12	B056	31	30000
15	C671	23	20000
18	E002	12	15000
15	E001	24	20000
18	B056	11	10000

Consider, Relation R(A, B, C, D, E, F) and the FDs as below,

A → BC , B → DC, D → EF

(i) The candidate Key is {AD} → {A, D, B, C, E, F} selected as primary key.
All attributes are partially dependent on primary key.
Hence, Relation R is **not in 2NF.**

(ii) The 2NF Relation Schema is,
R1 (A, B, C, D) with FDs A → BC, B → DC
R2 (D, E, F) with FDs D → EF

Minimizing Tuple Redundancy

The second normal form will avoid same tuples to be repeated in a table as it forces all non-key attributes must be full functionally depends on primary key of a relation. 2NF will create a new table for each partial key with all its dependent attributes.

Q. 14 Explain 3NF. May 2016

Ans. : Third Normal Form (3NF)

This normal form is given by E.F. Codd in 1971. This normal form introduced to minimize the transitive redundancy. To remove the data anomalies left in relational schema even after applying second normal form like transitive dependencies.

Definition

A relation is in 3NF, if it is in 2NF and all non-prime attributes of the relation are non-transitively dependent on the every key.

A relation R is in 3NF if all non prime attributes are,

1. Full functionally dependent on primary key.
2. Non-transitive dependent on every key.

A relational schema R is in 3NF, if non-trivial functional dependency X→A holds true where X is a superkey and A is a prime attribute.

Example

Consider an employee table with columns as shown in diagram, The relational schema **not in 2 NF** is represented as,

Consider an Employee table with following FDs,

Employee_Id → Ename, Salary, Department_Id

Department_Id → Dname

The state of Employee relational is,

Table 10.11 : Employee Table

Employee_Id	Ename	Salary	Department_Id	Dname
10	Mahesh	50000	C1	IT
12	Suresh	25000	E2	HR
15	Ganesh	26000	C1	IT
18	Mahesh	50000	E2	HR

{Employee_Id} → Ename, Salary, Department_Id, Dname

Therefore,

Candidate key {Employee_Id} is selected as primary key.

As all attributes in employee table are full functionally dependent on primary key. Therefore, **Relation R is in 2NF.**

Non-prime attributes Ename, Salary, Department_Id are non-transitively dependent on primary key. But, Dname attribute is transitively dependent on key. Therefore, **Relation R is not in 3NF.**

Employee_Id → Department_Id

Department_Id → Dname

To normalize above schema to 3NF we can decompose tables as ,

Employee (Employee_Id, Ename, Salary, Department_Id)

Employee_Id → Ename, Salary, Department_Id

Table 10.12 : 3NF Employee Table

Employee_Id	Ename	Salary	Department_id
10	Mahesh	50000	C1
12	Suresh	25000	E2
15	Ganesh	26000	C1
18	Mahesh	50000	E2

Department (Department_Id, Dname)

Employee_Id → Dname

Table 10.13 : 3NF Department Table

Department_Id	Dname
C1	IT
E2	HR

Consider, Relation R(A, B, C, D, E, F) and the FDs as below,

A → BC , B → D, D → EF

(i) The candidate Key is {A} → {A, D, B, C, E, F} selected as primary key. All attributes are full functionally dependent on primary key.

Hence, Relation R is **in 2NF.**

But, non-prime attributes B, D, E and F are transitively depends on key.

So, Relation R is **not in 3NF.**

(ii) The 3NF Relation Schema is,

R1 (A, B, C) with FDs A → BC

R2 (B, D) with FDs B → D

R3 (D, E, F) with FDs D → EF

Minimizing Group Redundancy

The third normal form will avoid repeating groups in same table as it forces all non-prime attributes must be non-transitively depends on key of a relation. 3NF will create a new table for each transitive attribute and its dependent attributes.

Q. 15 Describe BCNF in detail. Dec. 2015, Dec. 2016

Ans. : BCNF

This normal form is governed by Raymond F. Boyce and E.F. Codd in 1974. BCNF is more rigorous form of 3NF and every relation in BCNF is always in 3NF. The intention of Boyce-Codd Normal Form (BCNF) is that 3NF does not satisfactorily handle the case of overlapping candidate keys. If transitivity is present in prime attributes of relation may not be removed by 3NF.

Definition

A relation R is said to be in BCNF, if and only if every determinant is a candidate key. A relational schema is in BCNF, if a non-trivial functional dependency X→A is true then X is a superkey of relation R. In 3NF definition A should be prime attribute, which is not the case in BCNF definition.

Example

Consider an employee table in which employee can work in more than one department,

The relational schema **not in 2 NF** is represented as,

Consider an Employee table with following FDs,

Employee_Id → Ename, Salary, Department_Id

Department_Id → Dname

The state of Employee relational is,

Table 10.14 : Employee Table

Employee_Id	Ename	Department_Id	Dname	Dtype
10	Mahesh	C1	IT	Technical
12	Ganesh	E2	HR	Skill
12	Ganesh	C1	IT	Technical
10	Mahesh	E2	HR	Skill
13	Satish	E1	TS	Technical

Employee_Id → Ename

Department_Id → Dname, Dtype

Therefore,

Candidate key {Employee_Id, Department_Id} is selected as primary key. As no attribute in employee table is full functionally dependent on primary key. Therefore, **Relation R is not in 2NF.** All non-prime attribute are non-transitively dependent on primary key. Therefore, **Relation R is in 3NF.** To normalize above schema to **BCNF** we can decompose tables as ,

Employee (Employee_Id, Ename)

Employee_Id → Ename

The determinant Employee_Id is candidate key.

Table 10.15 : 3NF Employee Table

Employee_Id	Ename
10	Mahesh
12	Ganesh
13	Satish

Department (Department_Id, Dname, Dtype)

Department_Id → Dname, Dtype

The determinant Department_Id is candidate key.

Table 10.16 : 3NF Department Table

Department_Id	Dname	Dtype
C1	IT	Technical
E2	HR	Skill
E1	TS	Technical

Emp_Dept (Employee_Id, Department_Id)

Table 10.17 : 3NF Emp Dept Table

Employee_Id	Department_Id
10	C1
12	E2
12	C1
10	E2
13	E1

Minimizing Key Transitivity (Key redundancy)

The BCNF will produces a table for each functional dependency to make all determinants as key of relation. BCNF will create a new table for each transitive attribute and its dependent attributes.

Q. 16 Consider the following relation,
CAR-SALE (Car#, Date-sold, Salesman#, commission%, Discount-amt)
Assume that {Car#, Salesman#} is the primary key.
Additional dependencies are,
Date-sold → Discount_amt
Salesman# → commission%
Based on the given primary key, is this relation in 1NF, 2NF or 3NF ?
Why or why not ?
How would you successively normalize it completely ? Dec. 2014

Ans. :

CAR-SALE (Car#, Salesman#, Date-sold, commission%, Discount-amt)

Assuming {Car#, Salesman#} is the primary key

Therefore,

Car#, Salesman# → Date-sold, commission%, Discount-amt

Additinally,

Date-sold → Discount_amt

Salesman# → commission%

The above relation is in 1NF as all attributes of relation are atomic domains. The above relation is in 2NF as primary key is assumed and all non key attributes are full functionally depends on primary key. The above relation is in 3NF as all non prime attributes are non-transitively depending on primary key.

Noramlized Relation

CAR-SALE (Car#, Salesman#, Date-sold, commission%, Discount-amt)

Car#, Salesman# → Date-sold, commission%, Discount-amt

Date-sold → Discount_amt;

Salesman# → commission%

Q. 18 Consider a dependency diagram of relation R and normalize it up to third normal form. Dec. 2013

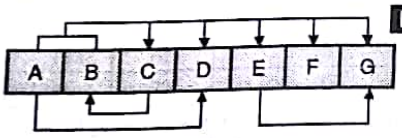


Fig. 10.1

Ans. :

Noramlized Relation R (A, B, C, D, E, F, G) with set of FDs

AB → CDEFG

C → B

A → D

E → G

Q. 19 Consider a dependency diagram of relation R and normalize it upto third normal form. Dec.2013

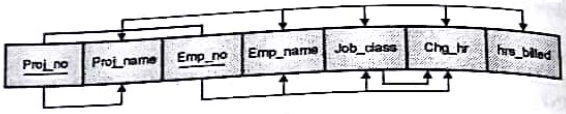


Fig. 10.2

Ans. :

Noramlized Relation,

Employees (Proj_no, Emp_no, Proj_name, Emp_name, Job_Class, Chg_Hr, Hrs_Billed)

With set of FDs

Proj_no, Emp_no → Proj_name, Emp_name, Job_Class, Chg_Hr, Hrs_Billed

Emp_no → Emp_name, Job_Class, Chg_Hr

Proj_no → Proj_name

Job_Class → Chg_Hr

Q. 20 Consider the following dependency diagram of relation R and normalize till 3NF form. May 2017

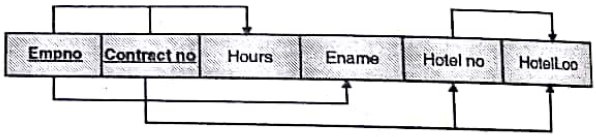


Fig. 10.3

Ans. :

Converting dependency diagram to dependencies,

Empno,Contactno -> Hours

Empno ->Ename

Contactno ->Hotelno,HotelLoc

Hotelno->HotelLoc

3NF schema is as given below,

Emp(Empno,Contactno,Hours); Empno,Contactno -> Hours

Emp_Details(Empno,Ename); Empno ->Ename

Hotel_Contact(Contactno,Hotelno,HotelLoc);

Contactno ->Hotelno,HotelLoc

Hotel(Hotelno,HotelLoc); Hotelno ->HotelLoc

Chapter 11 : Storage and Indexing

Q. 1 Describe various operations on file.

Ans. : File Operations

Operations are classified into two categories

a) **OPEN :** Reads the file for access
b) **FIND :** Searches for the first file record that satisfies a given condition and makes that record as current file record
c) **READ :** Reads the current file record into a variable
d) **INSERT :** Adds a new record into the file
e) **DELETE :** Removes the current record from the file
f) **MODIFY :** Changes the values of some fields in the current file record
g) **CLOSE :** Terminates access to the file

Q. 2 Describe various types of records in files.

Ans. : Types of Records in Files

Data in file organization is usually stored in the form of records. Records contain fields which have values of a particular entry in database.

There are various types of records,

1. **Fixed Length Record :** Consider a file of student records of the form :

```
Type t_student = record
    Student_name : char(22);
    Sid : char(10);
    fees : real;
end
```

If we assume that each character occupies one byte, an integer occupies 4 bytes, and a real 8 bytes, our deposit record is 40 bytes long. The simplest approach is to use the first 40 bytes for the first record, the next 40 bytes for the next record, and so on.

Problems

It is difficult to delete a record from such fix structure. Block size should be multiple of 40. It would then require two block accesses to read or write a record which is more then size 40.

2. **Variable Length Record :** Variable length records are required in system if Storage of multiple record types in a file, Record types that allow variable lengths for a single or multiple fields. It can be implemented as follows,

1. Byte string representation of variable length record.

SEIT	IT-101	45000	IT-201	40900	IT-301	70000	⊥
SECE	IT-102	35000	⊥				
TEIT	IT-103	42000	⊥				
ITCE	IT-104	47000	IT-204	45000	⊥		
BEIT	IT-105	43000	⊥				
BECE	IT-106	41000	⊥				

In method above attach a special end of record (⊥) symbol to the end of each record. In other method we store the record length at the beginning of each record.

Q. 3 Write a note on : File Organization.

Ans. : File Organizations

1. **Heap file organization :** Any record can be placed anywhere in the file where there is space for that record. There is no proper ordering of records in a file. This is file organization can be also called as 'pile file'. In this file organisation each new record is inserted at the end o f the file. To search any record, a linear (sequential) search through the file is necessary. This requires reading and searching all file blocks in worst case, and is hence quite expensive. In this file organisation insertion of record is quite efficient. Sorting the file is required for Reading the records in order of a particular field.
2. **Sequential file organization :** In A sequential file records are chained together with help of pointers to permit faster access in order of search key value. This file organisation is having efficient processing of records in sorted order based on some search key. Pointer points to next consecutive record in order inside file. Records are physically located in order of search key value. Deletion can be done with help of pointers put new record in an overflow block.

101	Mahesh	200	
102	Pravin	300	
103	Pankaj	400	
104	Bhavna	500	

Fig. 11.1

3. **Hashing file organization :** A hash function is computed with help of some attribute of every record. This hash function is used to identify location of record inside the file. Hashing technique done for disk files is also called External Hashing

Method of hashing

The file blocks are divided into M equal-sized *buckets* (0 to M-1). Each bucket corresponds to one disk block. One of the file fields is called to be as hash key of the file.

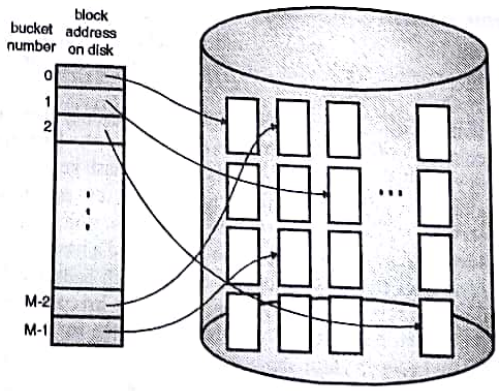


Fig. 11.2

The record with hash key value K is stored in bucket number i,

$$i = h(K),$$

Where h = *hash function*

Search is very efficient using the hash key. Collisions can happen when a new record hashes to a bucket which is already full. Such records are added to overflow bucket.

4. **Clustering file organization :** Records of multiple relations (tables) can be stored in the same file. Related records of the different relations are stored on the same block so that one I/O operation can fetch related records from any relations.
   Relation per file = 1
   Record type = fixed-length record
   Such file system is good for small size databases, which can reduce the code size. This Method combines tuples from multiple relations, but allows for efficient processing of the join. If the customer can have many accounts which cannot fit in one block, the remaining records can be stored on nearby blocks. Such file structure, called clustering.

Q. 4 Write a note on : Data Dictionary.

Ans. : Data Dictionary

1. A DBMS needs to maintain a description of all the data items that it contains. Like a RDBMS contains information about all relation, database objects, views and index present in it.
2. The DBMS should also maintains information about views (a definition of the view) to compute the tuples that belong in the view whenever the view is queried or called.
3. The multiple relations which stores information about data in database. Such tabes are created and maintained by the system itself. They are called as **catalog relations**.
4. The catalog relations are also called the **system catalog** or the **data dictionary.**
5. The system catalog is sometimes also referred to as **metadata (data about data)**
6. The information in the system catalog is generally used only for query optimization.

Q. 5 Explain concept of hashing and its type.

Ans. : Hashing Techniques

Hashing is a type of primary file organization which provides very fast access to records in file on certain search conditions. This type of file system is also called as a **hash file.**

Hashing is a technique used with file organization for accessing records comparatively faster without using indices. Hashing can also be used for creation of index-structure.

Terms Related to Hashing

(a) **Bucket :** A bucket is a unit of storage containing one or more records (a bucket is typically a disk block which stores records).

(b) **Hash Key :** The search condition in hashing must be an equality condition on a single field, called the hash field or key field of the file, this is also called the hash key.

(c) **Hash Function :** Uniform distribution of search values across a range of buckets. The bucket to which a value is assigned is determined by a function, called a hash function.
Ideal hash function is random, so that each bucket will have the same number of records assigned to it irrespective of the actual distribution of search-key values in the file.

(d) **Hash index :** Hash indices are always secondary indices. Hash index is one which organizes the search keys, with their associated record pointers, into a hash file structure.

(e) **Bucket Overflow :** Bucket overflow can occur because of insufficient buckets.
Skew in distribution of records. This can occur due to two reasons :
1. Multiple records have same search-key value
2. Chosen hash function produces non-uniform distribution of key values

Although the probability of bucket overflow can be reduced, it cannot be eliminated; it is handled by using overflow buckets.

Types of hashing

1. **Static Hashing :** Index schemes forces us for traverse an index structure while, hashing technique avoids this.

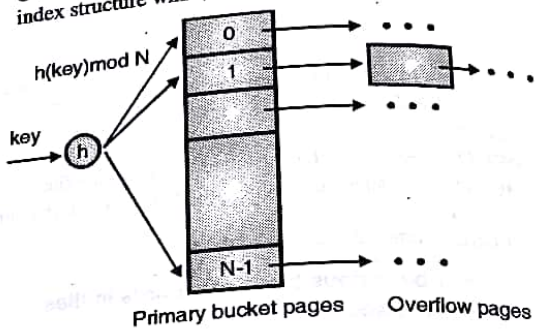


Fig. 11.4 : Concept of Hashing

The basic idea behind using hashing technique is to make use a hash function, which is used to map values in a search fields into a range of bucket numbers to find the page number on which a desired data is stored.

2. **Dynamic Hashing :** Good for database that grows and shrinks in size. Allows the hash function to be modified dynamically so problem of static hashing can be overcome.

3. **Extendable Hashing :** This is type of dynamic hashing. An extendible hash index consists of two parts:
   (a) **Buckets :** The buckets have a physical address on the disc Bucket contains a fixed number of records.
   (b) **Directory :** The directory indexes the buckets using some binary code. The directory consists of two parts: A binary code which is calculated from the hash function and A pointer to the bucket containing records matching the binary code.
   Two entries in directory may point to the same record.

Advantages

1. Performance of method does not degrade as file size increases
2. Stores the minimum number of buckets so optimizes storage space.
3. Number of buckets grows/shrinks dynamically

Disadvantages

1. The directory must be searched while looking for data.
2. The directory required to be stored.

Q. 6 Explain concept of B Trees and B+ Trees.

Ans. : B Trees

B Trees are multi-way trees means each node contains a set of keys and pointers. A minimum size of B Tree is with 4 keys and five pointers. B Trees are dynamic in nature in which the height of the tree grows or shrinks as new records are added and old records are deleted.

B + Trees

A B+ Tree combines features of ISAM and B Trees. A B+ tree is a balanced tree in which every path from the root to a leaf is of the same length. Each non-leaf node in the tree must have

between [n/2] and n children, where n is fixed for a particular tree. A B+ tree contains index pages and data pages. B+ Trees as well as B Trees use a "fill factor" to control the growth of data file. A 50% fill factor would be the minimum for any B+ or B tree.

Features similar to ISAM

1. The data always appear as leaf nodes in the tree.
2. The root node and intermediate nodes are always index pages.

Contrasting features to ISAM

Overflow pages are not used in case of B+ trees.

B + Tree Operations

(a) **Searching for node in B+ Tree :** A B+ Tree Starts search at root node then uses key comparisons to go to leaf through intermediate nodes.

(b) **Insertion of node in B+ Tree :** Value determines placement of record in a B+ tree. Generally, the leaf pages will be maintained in sequential order use a doubly linked list to connect each leaf page with its sibling page(s). Must consider three scenarios while inserting a record to a B+ tree.

Leaf Page	Index Page	Algorithm to follow
Not Full	Not Full	Place a given record in sorted position in the proper leaf page
Full	Not Full	1. Split the required leaf page into two. 2. Select middle key value from above values. 3. Rearrange all key values in the sorted order. 4. Left leaf page contains records with keys below the middle key. 5. Right leaf page contains records with keys equal to or greater than the middle key.
Full	Full	1. Split the required leaf page into two. 2. Left leaf page contains records with keys below the middle key. 3. Right leaf page contains records with keys equal to or greater than the middle key. 4. Split the required index page into two. 5. Left index page contains records with keys below the middle key. 6. Right index page contains records with keys equal to or greater than the middle key. 7. The middle key goes to the next (higher level) index. If the next level index page is full, continue splitting the index pages.

(c) **Deletion of node from B+ Tree**

A key value determines placement of record in a B+ tree.

To delete a record from B+ tree, must consider below given three scenarios and must consider a Fill Factor (FF) while deletion of records from data file,

Leaf Page	Index Page	Algorithm to follow
Above FF	Above FF	1. Delete the record from the leaf page. 2. Arrange keys in ascending order to fill void. If the key of the deleted record appears in the index page, use the next key to replace it.
Below FF	Above FF	Combine the leaf page and its sibling. Change the index page to reflect the change.
Below FF	Below FF	1. Combine the leaf page and its sibling. 2. Adjust the index page to reflect the change. Combine the index page with its sibling. Continue combining index pages until you reach a page with the correct fill factor or you reach the root page.

Advantages

i. B+ tree index automatically reorganizes itself in the case of insertions and deletions.

ii. Reorganization of entire file is not required to maintain performance.

Disadvantages

i. B+ tree index degrades performance as file grows. (As many overflow blocks get created).

ii. Periodic reorganization of entire file is required.

❑❑❑