

SYLLABUS
DATABASE MANAGEMENT SYSTEMS

Unit – I	Introduction to Databases and Transactions
	What is database system, purpose of database system, view of data, relational databases, database architecture, transaction management,
Unit- II	Data Models
	The importance of data models, Basic building blocks, Business rules, The evolution of data models, Degrees of data abstraction.
Unit-III	Database Design ,ER-Diagram and Unified Modeling Language
	Database design and ER Model:overview, ER-Model, Constraints, ER-Diagrams, ERD Issues, weak entity sets, Codd’s rules, Relational Schemas, Introduction to UML Relational database model: Logical view of data, keys, integrity rules. Relational Database design: features of good relational database design, atomic domain and Normalization (1NF, 2NF, 3NF, BCNF).
Unit- IV	Relational Algebra and Calculus
	Relational algebra: introduction, Selection and projection, set operations, renaming, Joins, Division, syntax, semantics. Operators, grouping and ungrouping, relational comparison. Calculus: Tuple relational calculus, Domain relational Calculus, calculus vs algebra, computational capabilities.
Unit- V	Constraints, Views and SQL
	What is constraints, types of constrains, Integrity constraints, Views: Introduction to views, data independence, security, updates on views, comparison between tables and views SQL: data definition, aggregate function, Null Values, nested sub queries, Joined relations. Triggers.
Unit-VI	Transaction management and Concurrency control
	Transaction management: ACID properties, serializability and concurrency control, Lock based concurrency control (2PL, Deadlocks),Time stamping methods, optimistic methods, database recovery management.

Books:

A Silberschatz, H Korth, S Sudarshan, "Database System and Concepts", fifth Edition McGraw-Hill ,
Rob, Coronel, "Database Systems", Seventh Edition, Cengage Learning.

Term Work and tutorial
Should contain 5 assignments and two class tests

Practical: Should contain minimum 8 experiments

Practicals

- 1) Design a Database and create required tables. For e.g. Bank, College Database
- 2) Apply the constraints like Primary Key , Foreign key, NOT NULL to the tables.
- 3) Write a sql statement for implementing ALTER,UPDATE and DELETE
- 4) Write the queries to implement the joins
- 5) Write the query for implementing the following functions: MAX(),MIN(),AVG(),COUNT()
- 6) Write the query to implement the concept of Intergrity constrains
- 7) Write the query to create the views
- 8) Perform the queries for triggers
- 9) Perform the following operation for demonstrating the insertion , updation and deletion using the referential integrity constraints
- 10) Write the query for creating the users and their role.

INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

Unit Structure

- 1.0 Objectives
- 1.1 Introduction
- 1.2 What is Database Management System
- 1.3 History of Database System
- 1.4 Purpose of Database System
- 1.5 Advantages and Disadvantages of Database System
- 1.6 Summary
- 1.7 Model Questions

1.0 OBJECTIVES

1.1 INTRODUCTION

- In today's world as the information technology has changed rapidly, many computing applications deal with large amounts of information regularly.
- As the end user applications has changed significantly in last few decades, there is a challenge to store the large amount of

information, retrieve and manage this information in timely manner.

- This can be achieved today by making use of services of Database Management System (DBMS).
- Today DBMS not only used to insert, update and delete the data stored in database.
- The job of DBMS system is to collect the data, give a systematic representation to it and also provides ways for the data to be modified or extracted by users or other programs.
- As the technology has grown rapidly in past four decades, today DBMS has gained its own importance because the data has brought online in the hands of end user through different computer networking.
- Our world is driven with a lot of exciting applications such as multimedia databases, live streaming of data, digital has made our life much easier to deal with data.

1.2 WHAT IS DATABASE MANAGEMENT SYSTEM

- A primary aim of the database system is to provide a *convenient* and *efficient* way to store and retrieve data stored in a database.
- A database is a computer generated software program which can be used to access the data stored in database in an organised manner.
- The term *database* is a structured collection of data stored which can be stored in digital form. Before the actual data is stored in the database, we should clearly specify the schema of the database and different techniques used to manipulate the data stored in a database.
- Database shouldn't only care about the insertion and modification of the data in the database. At times, it should also focus on how to protect the data stored in the database from unauthorised access.
- DBMS must provide efficient techniques in order to protect the data from accidental system crashes.

- If the data has to be shared among number of users there are highly chances that the data might not remain consistent because too many users might try to access it at same time and may try to change the value.
 - The DBMS must ensure that the chances of getting anomalous results when the data is used by more than one set of user. DBMS systems can be used extensively in the following fields
1. **Transportation:** DBMS system can be used for reservation or cancellation of tickets and can be also used to check for the schedules of incoming and outgoing flights.
 2. **Education:** DBMS system can be used by different universities to allow students take admission online, checking the status of vacant seats, enrolment system can be done computerized etc.
 3. **Banking:** DBMS system have completely changed the face of the banking sector. Few decades ago, the banking system was purely the paper based system have now transformed in keeping less of paper work.
 4. **Sales:** DBMS system allows the data to be stored in electronic format by making use of relational databases which allows the data to be stored in highly organised manner. This database allows the information such as information about the customers, products, sales, purchases etc to be stored in database.
 5. **Manufacturing:** DBMS system allows the user to store information about the production of good, the inventory details, the total number of orders, supply chain information in database so that it allows the decision makers to make critical decisions in timely manner.
 6. **Human Resource:** *DBMS has made the* life of HR team much better by allowing the team to compute tax deductions, employee wages, retrieving the details of the employees in faster manner as compared to traditional paper based approach which was time consuming.

Thus the growth of DBMS system has not only benefitted only to the customers or employees in an organisation but it has touched all the aspects of our lives.

1.3 HISTORY OF DATABASE SYSTEM

- The following are the historical perspective of DBMS system:
- In early 1960s, the first general purpose DBMS was designed by Charles Bachman at General Electric, which was later, called as IDS (*Integrated Data Store*).
- This IDS formed a groundwork for introduction of Network Data Model, which was later, standardized by CODASYL (*Conference on Data Systems Languages*).
- In late 1960s, IBM developed the IMS (*Information Management System*) which was widely used.
- This IMS formed groundwork for introduction of *Hierarchical Data Model*.
- By the joint venture of IBM and American Airlines, the *SABRE* system was launched which help the people to reserve the tickets.
- The new data representation framework was initially launched by Edgar Codd, called the *Relational data model*.
- Both Bachman and Edgar Codd were felicitated by ACM Turing Award in the year 1973 and 1981 for the outstanding contribution in the field of database system.
- With the time passed by, the DBMS system has matured significantly. As the development of relational DBMS has reached to larger users and the number of benefits from the same, it was widely accepted and many corporate houses started using this system for their day to day activities.
- As the popularity of relational DBMS started increasing, soon IBM, in early 1980s, has developed a SQL (*Structured Query Language*) for relational databases through their SYSTEM/R project.
- Later in late 1980s, SQL was standardized and the version SQL-1999 was adopted by ANSI (*American National Standard Institute*) and ISO (*International Organization for Standardization*).

- Many developments were done on DBMS since its birth. The concept of concurrent programming was introduced in DBMS system which allowed the users to run their programs concurrently.
- Later in 1999, James Gray was awarded by ACM Turing Award for his contribution towards Database Transaction Systems.
- The period between 1980 and 1990s saw many advances in the field of DBMS system. Several vendors try to build a system where more stress is given on complex analysis of data within an enterprise.
- Many complex data types such as images, texts etc, were launched during this period and many complex queries are given more emphasis.
- Over a period of time a new type of database system was brought in which was known as *Data Warehouse* system.
- By the introduction of ERP (*Enterprise Resource Planning*) and MRP (*Management Resource Planning*) packages, exciting new features were added to existing database system.
- Many other packages like *SAP, Baan, Oracle, PeopleSoft* which were user friendly, allowed the user to carry out their tasks easily.
- Most significant change in DBMS is through integration of DBMS with Internet which allowed DBMS to stored data accessed through Web Browser.
- It allowed the user to write their queries through Web forms, and the formatted output is tabulated through mark-up languages like HTML.
- As more and more data grown over a period of time, it is really challenging to maintain the consistency of data.
- Today we have multimedia databases, interactive video, streaming data, video libraries has completely change in the way in which data is stored which allowed the company to simplify their decision making process.

1.4 PURPOSE OF DATABASE SYSTEM

Let us understand the need of database system with help of following example.

- To see why database management system is necessary, let us look at a typical “file processing system” supported by conventional operating system.
- The application is a saving bank:
 - Saving account and customer records are kept in permanent system files.
 - Application programs are written to manipulate files to perform following task.
 1. To debit or credit an account.
 2. To add a new account
 3. To find the account balance.
 4. To generate the monthly statements
- System programmers wrote these application programs to meet the needs of the bank.
- The system must be developed as per the following procedure:
 1. As per the necessity, the new application programs must be needed as and when it is needed.
 2. As per the requirement, the new permanent files are created.
 3. After certain interval of time, the files may be stored in a different format.
 4. Many application programmers have written their respective applications programs in different languages.
- File system has several disadvantages and the following problems are associated with file system:
 - 1. Data redundancy and inconsistency**
 - The major problem with file processing system is that it maintains several versions of same file i.e.; duplication of data is possible at multiple places.

- Also there are several copies of files are stored, if any one of the file is changed, the different versions of same file may not be updated which leads to inconsistency of data.

2. Difficulty in accessing the data

- Consider the airline reservation system. If the senior management of company wants to access the information of all its customers who are living in the same postal code, it has to be done manually because current file processing system does not allow the user to obtain this information.
- So in the above case, there are two options. Either the application programmer has to write a new application program to satisfy the unusual request or could get this information manually.
- In former case, it doesn't guarantee that the same query will be asked and same application program would be used in future.
- If a query changes, a new application program should be written to get the needed information.

3. Data isolation

- One of the major problems with the file system is that the data is scattered and stored in multiple locations and in different formats.
- Hence in order to retrieve the needed information from multiple location and in different formats is a very difficult to proceed with the help of application program.

4. Concurrent access anomalies

- In order to speed up the performance of the system and faster response to applications, many systems allow the user to update the data concurrently.
- Suppose two users located at different locations wants to book the tickets, there might be situation that both of the people will be given the same seat because the data is stored in multiple locations and both of them will be given a seat from individual copy of the data.

- Therefore there should be some protection mechanism to avoid this concurrent updates.

5. Security problems

- Every user in this system should be able to access the data which he is allowed to access and not all the data.
- For example, the salesperson in an organization should be allowed to access the data related to him and should not be allowed to access data which is used HR team or finance department in an organization.
- If the new constraints are added to avoid this kind of unauthorized access, enforcing these constraints is difficult because the existing application programs are added to the system in an adhoc manner.

6. Integrity problems

- Data stored in the database should be allowed to satisfy certain constraint checking.
- For eg, before adding a new employee in the Employee table, if we check the age of the employee and if we apply constraint such that only those employee whose age is greater than 18 years should be allowed to enter in the table which means that before the new data is inserted the age of the employee should be calculated.
- When a new constraint such as one which is discussed above is added, it becomes difficult to change the existing programs to enforce the new constraints.

7. Atomicity problems

- Every application system is assumed to fail at some point in near future.
- In many applications, if the system fails, the data should be rolled back to the state before the failure occurs.
- Consider the customer is withdrawing some cash from the ATM machine from his own account and if the failure happens in the system, it should not happen that the amount is deducted from customer account but the customer is not getting any cash from the machine.

- In simple word the withdrawn should be atomic- it must be happen in its entirely or not at all.
- Another disadvantage with file processing system is that it becomes difficult to ensure atomicity.

1.5 ADVANTAGES AND DISADVANTAGES OF DATABASE SYSTEMS

1.5.1. Advantages of Database Systems

The DBMS (Database Management System) is preferred over the conventional file processing system due to the following advantages:

1. Controlling Data Redundancy

- In the conventional file processing system, every user group maintains its own files for handling its data files. This may lead to
 - Duplication of same data in different files.
 - Wastage of storage space, since duplicated data is stored.
 - Errors may be generated due to updation of the same data in different files.
 - Time in entering data again and again is wasted.
 - Computer Resources are needlessly used.
 - It is very difficult to combine information.

2. Elimination of Inconsistency

- In the file processing system information is duplicated throughout the system. So changes made in one file may be necessary be carried over to another file. This may lead to inconsistent data. So we need to remove this duplication of data in multiple file to eliminate inconsistency.
- Let us consider the following example of student.
- Imagine that a particular student has opted for Embedded system as one of the elective subject in Sem –V for TYBScIT Sem V examination while filling up the examination form.
- If, after getting the hall ticket the student realize that rather than expecting Embedded system as the choice of elective

subject in the hall ticket, if some other subject is highlighted, it means that the data for that student has not correctly inserted in the database.

- To avoid the above problem, there is a need to have a centralized database in order to have this conflicting information.
- On centralizing the data base the duplication will be controlled and hence inconsistency will be removed.

3. Better service to the users

- A DBMS is often used to provide better services to the users. In conventional system, availability of information is often poor, since it normally difficult to obtain information in a timely manner because our existing systems are not capable to produce the same.
- Once several conventional systems are combined to form one centralized database, the availability of information and its updateness is likely to improve since the data can now be shared and DBMS makes it easy to respond to anticipated information requests.
- Centralizing the data in the database also means that user can obtain new and combined information easily that would have been impossible to obtain otherwise.
- Also use of DBMS should allow users that don't know programming to interact with the data more easily, unlike file processing system where the programmer may need to write new programs to meet every new demand.

4. Flexibility of the System is Improved

- Since changes are often necessary to the contents of the data stored in any system, these changes are made more easily in a centralized database than in a conventional system.
- Applications programs need not to be changed on changing the data in the database.

5. Integrity can be improved

- Since data of the organization using database approach is centralized and would be used by a number of users at a time, it is essential to enforce integrity-constraints.
- In the conventional systems because the data is duplicated in multiple files so updating or changes may sometimes lead to entry of incorrect data in some files wherever it is applicable.
- **For example:** - The example of Hall Ticket Generation system that we have already discussed, since multiple files are to maintained, as sometimes you may enter a value for subject which may not exist. Suppose Elective Subjects can have values (Embedded Systems, Advanced Java, Web Designing etc) but we enter a value 'Mathematics -I' for it, it may lead to database inconsistency.
- Even if we centralized the database it may still contain incorrect data. For example: -
 - Salary of full time clerk may be entered as Rs. 1500 rather than Rs. 4500.
 - A student may be shown to have borrowed library books but has no enrollment.
- The above problems can be avoided by defining the validation procedures whenever any update operation is attempted.

6. Standards can be enforced

- Standards are easier to enforce in database systems because all the data in database is access through centralized DBMS.
- Here standards may relate to the naming of data, structure of data, format of the data etc.
- Standardizing stored data formats is usually desirable for the purpose of data interchange or migration between systems.

7. Security can be improved

- In conventional systems, applications are developed in an adhoc manner.

- Often different system of an organization would access different components of the operational data, in such an environment enforcing security can be quite difficult.
- Setting up of a database makes it easier to enforce security restrictions since data is now centralized.
- It is easier to control who has access to what parts of the database. Different checks can be established for each type of access (retrieve, modify, delete etc.) to each piece of information in the database.
- Consider an example of banking in which the employee at different levels may be given access to different types of data in the database.
- For example, a clerk may be given the authority to know only the names of all the customers who have a loan in bank but not the details of each loan the customer may have.
- This can be accomplished by giving the privileges to each employee.

8. Organization's requirement can be easily identified

- All organizations have sections and departments and each of these units often consider the work of their unit as the most important and therefore consider their need as the most important.
- Once a database has been setup with centralized control, it will be necessary to identify organization's requirement and to balance the needs of the competition units.
- So it may become necessary to ignore some requests for information if they conflict with higher priority need of the organization.
- It is the responsibility of the DBA (Database Administrator) to structure the database system to provide the overall service that is best for an organization.
- For example, a DBA must choose best file Structure and access method to give fast response for the high critical applications as compared to less critical applications.

9. Data Model must be developed

- Perhaps the most important advantage of setting up of database system is the requirement that an overall data model for an organization be build. In conventional systems, it is more likely that files will be designed as per need of particular applications demand.
- The overall view is often not considered. Building an overall view of an organization's data is usual cost effective in the long terms.

10. Provides backup and Recovery

- Centralizing a database provides the schemes such as recovery and backups from the failures including disk crash, power failures, software errors which may help the database to recover from the inconsistent state to the state that existed prior to the occurrence of the failure, though methods are very complex.

1.5.1. Disadvantages of Database Systems

The following are the disadvantages of Database Systems

1. Database Complexity

The design of the database system is complex, difficult and is very time consuming task to perform.

2. Substantial hardware and software start-up costs

Huge amount of investment is needed to setup the required hardware and the softwares needed to run those applications.

3. Damage to database affects virtually all applications programs

If one part of the database is corrupted or damaged because of the hardware or software failure, since we don't have many versions of the file, all the application programs which are dependent on this database are implicitly affected.

4. Extensive conversion costs in moving form a file-based system to a database system

If you are currently working on file based system and need to upgrade it to database system, then large amount of cost is

incurred in purchasing different tools, adopting different techniques as per the requirement.

5. Initial training required for all programmers and user.

Large amount of human efforts, the time and cost is needed to train the end users and application programmers in order to get used to the database systems.

1.6 SUMMARY

- A Database Management system is the group of interrelated data and a set of programs to access that data.
- DBMS must provide efficient techniques in order to protect the data from accidental system crashes.
- A primary aim of the database system is to provide a *convenient* and *efficient* way to store and retrieve data stored in a database.
- The DBMS must ensure that the chances of getting anomalous results when the data is used by more than one set of user.
- DBMS system can be used in the fields such as transportation, education, banking, sales, manufacturing, human resource etc.
- The first general purpose DBMS was developed by Charles Bachman in early 1960s.
- Edgar Codd has suggested a new data representation technique known as relational model.
- SQL 1999 was standardized by ANSI and ISO in late 1980s.
- Different disadvantages of file system with respect to database system are listed below
 - 1 Data redundancy and inconsistency.
 - 2 Difficulty in accessing data
 - 3 Data isolation
 - 4 Concurrent data anomalies
 - 5 Security problems
 - 6 Integrity problems
 - 7 Atomicity problems
- The following are the advantages of DBMS
 1. Controlling Data redundancy
 2. Elimination of inconsistency
 3. Better services to the users
 4. Better flexibility
 5. Integrity is improved

6. Standards can be enforced.
7. Security can be improved etc.

1.7 MODEL QUESTIONS

1. What is the purpose of building a DBMS system?
2. Explain the history of Database system
3. What is the database system? Explain it with its advantages and disadvantages
4. Compare between File systems and database systems
5. What are the limitations of File processing systems? How that can be solved by using Database system?



INTRODUCTION TO RELATIONAL DATABASE MANAGEMENT SYSTEM

Unit Structure

- 2.0 Objectives
- 2.1 Introduction to RDBMS
- 2.2 The Relational Model
- 2.3 Introduction to SQL
- 2.4 Working with relations of RDBMS
- 2.5 Advantages and Disadvantages of Relational Database System
- 2.6 Summary
- 2.7 Model Questions

1.0 OBJECTIVES

2.1 INTRODUCTION TO RELATIONAL DATABASE MANAGEMENT SYSTEM

- A relational DBMS is special system software that is used to manage the organization, storage, access, security and integrity of data.
- This specialized software allows application systems to focus on the user interface, data validation and screen navigation.
- When there is a need to insert, modify, delete or display data, the application system simply makes a "call" to the RDBMS.

2

- Although there are many different types of database management systems, relational databases are by far the most common.
- Other types include hierarchical databases and network databases.
- Although database management systems have been around since the 1960s, relational databases didn't become popular until the 1980s when the power of the computer skyrocketed and it became feasible to store data in sets of related tables and provided real-time data access.
- A relational DBMS stores information in a set of "tables", each of which has a unique identifier or "primary key".
- The tables are then related to one another using "foreign keys". A foreign key is simply the primary key in a different table.
- RDBMS are widely used in real life applications such as:
 1. **Airlines:** It can be used to keep the status of the flights and schedules and for reservations and cancellation of tickets.
 2. **Banking:** It is useful in storing the customer information, account details, loan details and banking transactions.
 3. **Universities:** It is useful in storing the student information, course registrations, grades etc.

2.2 THE RELATIONAL MODEL

- The relational model is a collection of relations required to build a database. Informally, each relation resembles a table of values or, to some extent, a "flat" file of records.
- In relational model, each row in the table consists of a set of related data values.
- In this model, each row in the table shares some reality which corresponds to the real world entity or relationship.
- Every table and the columns present in the table are given a unique table name and column names which can be used to extract the relevant values from the tables.
- Note that in the given database, no two tables can have the same name but across the database the table name can be the same.
- Similarly, within the same table, no two columns can have the same column name. A duplicate column name can be given across the table.

- Before we proceed with more details on relational model, consider the following example. We will also define various terminologies associated with relational model.
- Consider the IDOLSYIT table given below

Table name: IDOLSYIT

Relation

Domain

Attributes

S_ID (Int)	S_NAME (String)	Contact no(Int)	Email (String)
1001	Amit	5265389125	amit@gmail.com
1002	Akash	1452639872	akash@hotmail.com
1003	Devang	4256398712	devang@yahoo.com
1004	Ritesh	4523689713	ritesh@rediffmail.com
1005	Rakesh	9823657412	Rakesh@gmail.com

Tuples

Fig 2.1 : The relationship between domains, attribute, tuples and Relation

- Now let us understand the domain, tuples and attributes in brief
- A row of records in the given table is called as tuple. In the above example, the individual records for students starting with 1000 to 1005.
- In the above example, the individual columns are called as attributes of the system.
- The table itself is called as the relation.
- The data type describing the types of values that can appear in each column is called a domain.
- In the above example, if we define the relation schema it would look like this

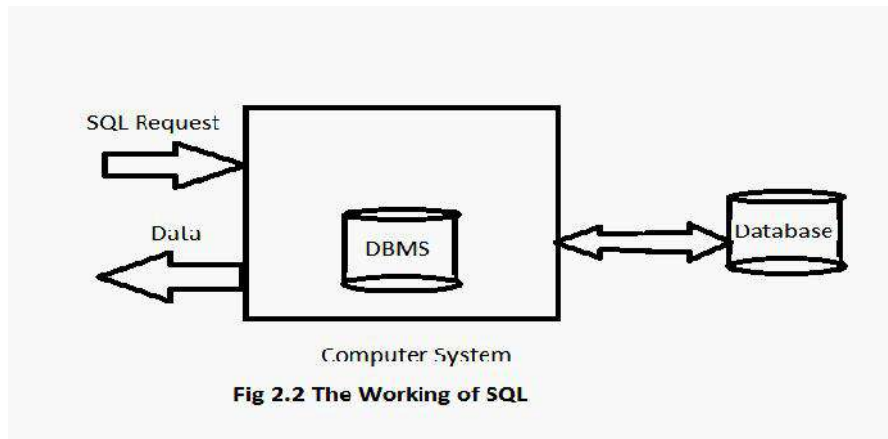
IDOLSYIT(S_ID :integer, S_NAME : String, contact no : integer, email : String)

- This says, for instance, that the field named *sid* has a domain named string.
- We now turn to the instances of a relation. An instance of a relation is a set of tuples, also called records, in which each tuple has the same number of fields as the relation schema.
- A relation instance can be thought of as a *table* in which each tuple is a *row*, and all rows have the same number of fields.

- We need to understand the different types of keys associated with relational databases as follows
- Primary key: In every relational database, every table has a particular column or set of columns whose value uniquely identify each row in the table. Such a column is called the primary key of the table.
- In our IDOLSYIT table, we can call S_ID as a primary key because it can uniquely define the values from this table.
- The primary key has a different unique value for each row in the table, so no two rows of a table with a primary key are exact duplicates of one another.
- In a table, if every row in a given table is different from all other rows is called the relation in mathematical terms.
- The term relational databases come because relations are the base of a relational model.
- A column in one table whose value matches with the primary key of another table is called as a foreign key of the table.

2.3 INTRODUCTION TO SQL

- SQL is a standard computer database programming language and its popularity has explored since past two decades.
- It is portable language which supports right from mainframe systems to personal computers and even to hand held devices.
- Today most of company's software products lie on SQL for its data management and SQL is the nucleus of database products from Microsoft and Oracle, two of the largest software companies in the world.
- The journey of SQL is a tremendous right from the beginning as an IBM research project; SQL has become a powerful market force.
- SQL is a vehicle for structuring, organizing, managing and retrieving data stored in the database.
- The name "SQL" is an abbreviation for *Structured Query Language*.
- It acts as an interpreter which allows the user to interact directly with database through computer language.
- The figure below shows how actually SQL works with databases



- In the above system, the computer system has a database which stores all the needed information.
- If the above database is for a company, it might store the information of manufacturing, finance, human resource, inventory, payroll etc.
- On the personal computer, the client must have created a database to store information such a list of people, their names, contact details etc. or data extracted from the larger computer system.
- When there is a need to retrieve the data stored in database, we take the help of SQL which allows the user to design queries based on user's choice which will retrieve the needed information from the database.
- The SQL then makes a request which is then processed by DBMS, retrieves the requested data and it returns the data back to the user.
- This process of requesting data from a database and receiving back the results is called a database *query*—hence the name Structured *Query* Language.
- There are various roles which are played by SQL. Some of them are discussed below

1. SQL is an interactive query language

SQL provides a very user friendly, easy to use tool which allows the user to write the typical SQL commands in order to retrieve the data from the database.

2. SQL is a database programming language.

Through the use of database utility programs, programmers write SQL commands in their own applications to retrieve the data stored in database.

3. SQL is a database administration language.

It allows the database administrators to define database structures and can also control the access to the stored data.

4. SQL is a client/server language.

In the client server architecture, the client programs uses SQL to communicate through a network to access the shared data stored in database.

5. SQL is an Internet data access language.

Since SQL is a standard language, many Internet web servers makes use of SQL to interact with company data and Internet application servers for accessing company-wide databases.

6. SQL is a distributed database language.

Many DDBMS (Distributed Database Management System) uses SQL to distribute the data across many connected computer systems. The DBMS software running on the local systems makes use of SQL to communicate with other systems by sending request for data access.

7. SQL is a database gateway language.

SQL is most of the time used as a gateway which allows one brand of DBMS to communicate with the other brands.

2.4 WORKING WITH RELATIONS OF RDBMS

- This section highlights how to create, modify or delete relations which may exist in relational model. This can understood by the following SQL statements
 1. Creating Relations(Create Table statement)
 2. Modifying Relations (Alter table statement)
 3. Integrity constraints over the relation

1. Creating a Relation (create table statement)

- The CREATE TABLE statement defines a new table (Relation) in the database and prepares it to accept data.
- For example, if we want to create a new table IDOLTYIT, the table is created as follows

```
Create table IDOLTYIT (S_ID integer not null, S_name
varchar(25) not null, contactno integer not null, email
varchar(30) not null).
```

- When the user is creating the above table, the user now become the owner of the newly created table, which is given the name specified in the above statement.
- Note that the table name must be a legal SQL name, and it must not conflict with any of the existing tables.
- A slightly complex create table is discussed below which allows to create a new table and also to set up the relationship between different tables

Create table NEWORDERS (ONUM INTEGER NOT NULL,

```

ODATE DATE NOT NULL,
CUST INTEGER NOT NULL,
REP INTEGER,
MFR CHAR (3) NOT NULL,
PRODUCT CHAR (5) NOT NULL,
QTY INTEGER NOT NULL,
AMT MONEY NOT NULL,
PRIMARY KEY (ONUM),
CONSTRAINT PLACEDBY
FOREIGN KEY (CUST)
REFERENCES
NEWCUSTOMERS
ON DELETE CASCADE,
CONSTRAINT TAKENBY
FOREIGN KEY (REP)
REFERENCES NEWSALESREPS
ON DELETE SET NULL,
CONSTRAINT ISFOR
FOREIGN KEY (MFR, PRODUCT)
REFERENCES NEWPRODUCTS
ON DELETE RESTRICT)

```

- In the above example, onum denotes the order number for every order taken by customer. Since it is having only unique values and no duplicates are allowed here, we can make this as primary key.
- In the above example, the column name CUST has made a foreign key which is currently referencing NEWCUSTOMERS table. REP column is made a foreign key referencing NEWSALESREPS table and MFR, PRODUCT is made the composite foreign key referencing the NEWPRODUCTS table.
- By setting the primary- foreign key relationship (Parent- Child Relationship), it allows the data to flow easily between the set of tables define in the Create table statement.

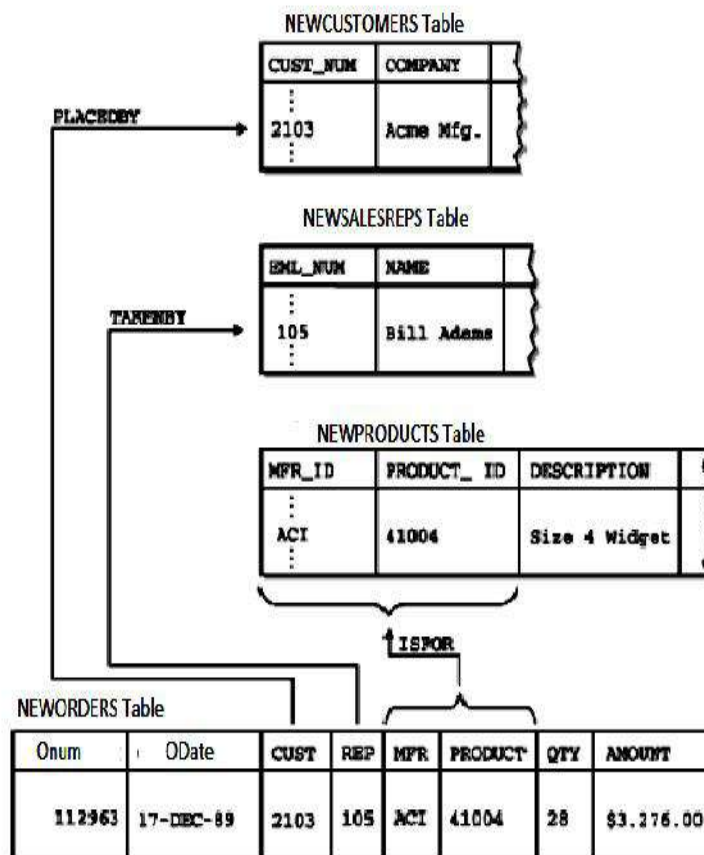


Fig 2.3 Relationship between table by making use of Create Table Statement

2. Modifying a relation (Alter table statement)

- After the table is ready, at times user feels the need to store additional information about the entries in the table.
- The alter table allows the user to change or modify the relation (schema) of the table which is already created by a Create table syntax.
- The alter table statement allows the user to do the following:
 1. Add a column definition to the table.
 2. Drop a column from the table.
 3. Change the default value to the table.
 4. Add or drop primary key for the table.
 5. Add or drop the foreign key for a table.
 6. Add or drop the uniqueness constraint for a table.
 7. Add or drop check constraint for a table.
- Some of the examples are discussed below
 1. Alter Table IDOLSYIT
Add Subject char (15)

- In the above example, an existing table of IDOLSYIT is modified with a new column is added as subject which was not there earlier.

2. Alter Table IDOLTYIT
Drop Email

- In the above example, an existing table of IDOLTYIT is modified with a existing column is removed known as email which was earlier present in the table.

3. Alter Table NEWOFFICES
Add Constraint Myoffices
Foreign key (NRegion)
References Regions

- In the above example, the NRegion column in the NEWOFFICES table is made a foreign key for the newly created Regions Table.

4. Alter Table NEWSALESREPS
DROP Constraint NWORKSIN
Foreign Key (NewRepOffice)
References NEWOFFICES

Alter Table NEWOFFICES
Drop Primary Key (Office)

- 3. In the above example, the primary key of the NEWOFFICES table has been changed. Before changing it, first we need to drop the reference to foreign key and primary key and then we need to select a new column from the table and should be made as a primary key.

3. Integrity constraints over the Relation

- The term *data integrity* refers to the correctness and completeness of the data in a database. When the contents of a database are modified with the INSERT, DELETE, or UPDATE statements, the integrity of the stored data can be lost in many different ways.
- One of the goals of RDBMS is to preserve the integrity of stored data to larger extent.
- To preserve the consistency and correctness of its stored data, a relational DBMS typically imposes one or more data integrity constraints.
- These constraints restrict the data values that can be inserted into the database or created by a database update.

- Several different types of data integrity constraints are commonly found in relational databases, includes the following
1. **Required data checking**
 - There are instances when some columns in a database must contain a valid data value in every row; they are not allowed to contain missing or NULL values.
 - In the sample database, every order must have an associated customer who placed the order. The DBMS can be asked to prevent NULL values in this column.
 2. **Validity checking**
 - Every column in a database has a domain, a set of data values that are legal for that column. The DBMS can be asked to prevent other data values in these columns.
 3. **Entity integrity**
 - The primary key of a table must contain a unique value in each row, which is different from the values in all other rows.
 - Duplicate values are illegal, because they wouldn't allow the database to distinguish one entity from another. The DBMS can be forced to enforce this unique values constraint.
 4. **Referential integrity**
 - A foreign key in a relational database links each row in the child table containing the foreign key to the row of the parent table containing the matching primary key value.
 - The DBMS can be asked to enforce this foreign key/primary key constraint.
 5. **Other data relationships**
 - The real-world situation modelled by a database will often have additional constraints that govern the legal data values that may appear in the database.
 - The DBMS can be asked to check modifications to the tables to make sure that their values are constrained in this way.

6. Business rules

- Updates to a database may be constrained by business rules governing the real-world transactions that are represented by the updates.
- For example, there might be a business rule such as the new employee should be added only if the age of the employee is between 18 to 35 years.

2.5 ADVANTAGES AND DISADVANTAGES OF RDBMS

ADVANTAGES**1. Simple data Structures**

- By storing the data in table format, it becomes easier for the users to understand the structure of database and use it.
- RDBMS provides data access using a natural structure and organization of the data.
- When the users are writing a queries, database queries can search any columns for any matching entries.

2. Multi-user database access monitoring

- RDBMS allows the multiple database users to access a database simultaneously.
- By taking advantage of services of transaction management and locking, it allows the user to access the data without being changed, prevents collisions between two users updating the same data, and keeps users from accessing partially updated records.

3. Well defined privileges

- Authorization and privilege control features in an RDBMS allow the database administrator to restrict access to authorized users, and grant privileges to individual users based on the types of database tasks they need to perform.
- Authorization can be defined based on the remote client IP address in combination with user authorization, restricting access to specific external computer systems.

4. Network Access

- RDBMSs provide access to the database through a server daemon, a specialized software program that listens for requests on a network, and allows database clients to connect to and use the database.
- Users do not need to be able to log in to the physical computer system to use the database, providing convenience for the users and a layer of security for the database. Network access allows developers to build desktop tools and Web applications to interact with databases.

5. Speed

- The relational database model is not the fastest data structure. RDBMS advantages, such as simplicity, make the slower speed a fair trade-off.
- Optimizations built into an RDBMS, and the design of the databases, enhance performance, allowing RDBMSs to perform more than fast enough for most applications and data sets.
- Improvements in technology, increasing processor speeds and decreasing memory and storage costs allow systems administrators to build incredibly fast systems that can overcome any database performance shortcomings.

6. Maintenance

- RDBMS feature maintenance utilities that provide database administrators with tools to easily maintain, test, repair and back up the databases housed in the system.
- Many of the functions can be automated using built-in automation in the RDBMS, or automation tools available on the operating system.

7. Language

- RDBMSs support a generic language called "Structured Query Language" (SQL).
- The SQL syntax is simple, and the language uses standard English language keywords and phrasing, making it fairly intuitive and easy to learn.

2.6 SUMMARY

- A relational DBMS is special system software that is used to manage the organization, storage, access, security and integrity of data.
- A relational DBMS stores information in a set of "tables", each of which has a unique identifier or "primary key".
- RDBMS is the organization of data stored in rows and columns.
- *Tuples* are the rows of the records in the given table.
- *Attributes* are the column headers of the table.
- The data type describing the types of values that can appear in each column is called a *domain*.
- In every relational database, every table has a particular column or set of columns whose value uniquely identify each row in the table. Such a column is called the primary key of the table.
- A column in one table whose value matches with the primary key of another table is called as a foreign key of the table.
- SQL is a vehicle for structuring, organizing, managing and retrieving data stored in the database.
- SQL plays various roles. Some of them are listed below
 1. SQL is an interactive query language
 2. SQL is a database programming language.
 3. SQL is a database administration language.
 4. SQL is an Internet data access language.
 5. SQL is a client/server language.
 6. SQL is a distributed database language.
 7. SQL is a database gateway language.
- The CREATE TABLE statement defines a new table (Relation) in the database and prepares it to accept data.
- The ALTER table allows the user to change or modify the relation (schema) of the table which is already created by a Create table syntax.
- The term *data integrity* refers to the correctness and completeness of the data in a database. When the contents of a database are modified with the INSERT, DELETE, or UPDATE statements, the integrity of the stored data can be lost in many different ways.
- Various Integrity checking discussed such as
 1. Required data checking
 2. Validity checking

3. Entity integrity
4. Referential integrity
5. Other data relationships
6. Business rules

2.7. MODEL QUESTIONS

1. What is RDBMS? Explain the need for RDBMS.
2. Explain the relational model with suitable example.
3. Define the following terms
 - a. Tuple
 - b. Attribute
 - c. Domain
 - d. Primary Key
 - e. Foreign Key
4. Write in detail about SQL.
5. Explain the various role of SQL
6. Explain how to create and modify the relations of RDBMS
7. Explain the various advantages of SQL.



INTRODUCTION TO DATABASE STRUCTURE

Unit Structure

- 3.0 Objectives
- 3.1 Levels of abstraction in DBMS
- 3.2 View of data
- 3.3 Role of Database users
- 3.4 Role of database administrators
- 3.5 Transaction Management
- 3.6 Database Structure
- 3.7 Summary
- 3.8 Model Questions

3.0 OBJECTIVES

Introduction

- In traditional system, each collection of application programs had its own independent master file. The duplication of data over master files could lead to inconsistent data.
- In early days, efforts were discovered to use a common master file for a number of application programs resulted in problems of integrity and security.

- The production of new application programs could require amendments to existing application programs, resulting in '*unproductive maintenance*'.
- Data structuring techniques, developed to exploit random access storage devices, increased the complexity of the insert, delete and update operations on data.
- As a first step towards a DBMS, packages of subroutines were introduced to reduce programmer effort in maintaining these data structures.
- However, the use of these packages still requires knowledge of the physical organization of the data.
- A database system is a computer-based system to record and maintain information. The information concerned can be anything of significance to the organisation for whose use it is intended.
- The database can hold a variety of different things. The database concepts are divided into two concepts:
 1. Schema
 2. Data
- The schema is the structure of the database and the data is the facts of the database.
- Consider our Salesperson database where we are storing the facts of the salespeople working in an organization.
- Such facts could include salesperson name, address, date of birth, and salary. In a database all the information on all salespeople would be held in a single storage "container", called a *table*.
- This table is a tabular object like a spreadsheet page, with different salespeople as the rows, and the facts (e.g. their names) as columns. Let us call this table Salesperson, and it could look something like:

Salesperson

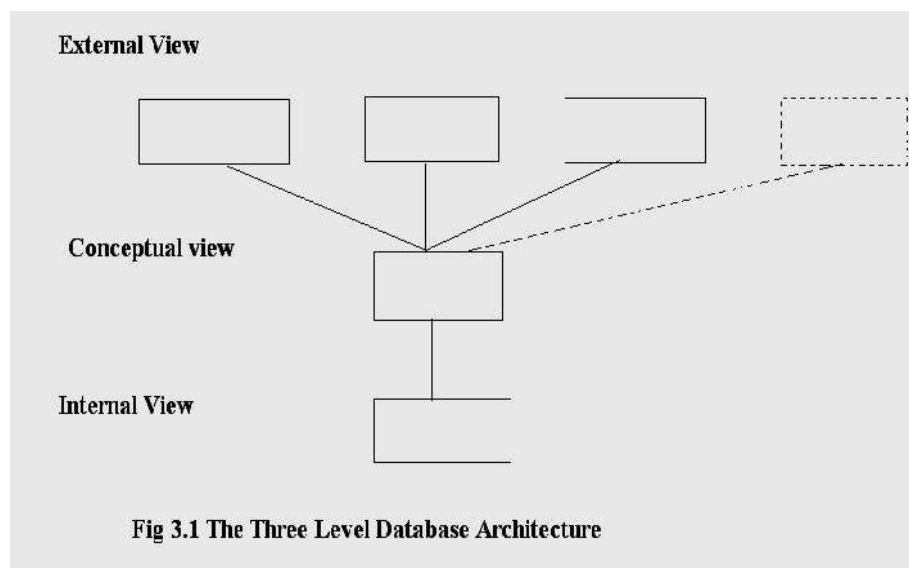
Name	Address	Date of Birth	Salary
Rakesh	M.G. Road	12/12/1960	11000
Dinesh	CST Road	15/11/1978	25000
Sudhir	JN Road	14/02/1985	15000

- From the above example, the schema would define that Salesperson table has four components, "Name", "Address", "Date of Birth", and "Salary".

- At times as a database administrator, you want to protect user from accidentally entering wrong data. For example, you don't want the user to enter the name in date of birth field in database.
- Protecting the database against rubbish data is one of the most important database design steps, and is what much of this course is about. From what we know about the facts, we can say things like:
 1. NAME is a string, and needs to hold at least 12 characters.
 2. ADDRESS is a string, and needs to hold at least 12 characters.
 3. DOB is a date... The company forbids people over 100 years old or younger than 18 years old working for them.
 4. SALARY is a number. It must be greater than zero.

3.1 LEVELS OF ABSTRACTION IN DBMS

- Database management can be defined in the way in which they use their data dictionary.
- The data dictionary contains logical descriptions of the data and its relationships, physical information about data storage, and usually information on users on users and privileges.
- Data dictionaries are helpful for all human users, especially the database administrator, as well as invaluable to the application programs and report generators that might access the database.



- The three levels of database architecture are
 1. External Level: It is concerned with the way individual user observes the data.
 2. Conceptual Level: It can be regarded as a community user view a formal description of data of interest to the organisation, independent of any storage considerations.
 3. Internal Level: It is concerned with the way in which the data is actually used.
- Let us discuss this three levels in more detail

1. External Level

- A user is anyone who needs to access some portion of the data from the database.
- They may range from application programmers to casual users with complex adhoc queries.
- Each user may use the language according to its own choice.
- The application programmer may use a high level language (e.g. COBOL) while the casual user will probably use a query language.
- Regardless of the language used, it will include a data sub-language (DSL) which is that subset of the language which is concerned with storage and retrieval of information in the database and may or may not be apparent to the user.
- A DSL is a combination of two languages:
 1. A data definition language (DDL) which provides for the definition or description of database objects
 2. A data manipulation language (DML) which supports the manipulation or processing of database objects.
- Each user sees the data in terms of an external view which is defined by an external schema, consisting basically of descriptions of each of the various types of external record in that external view, and also a definition of the mapping between the external schema and the underlying conceptual schema.

2. Conceptual Level

- It defines the logical definition of the database.
- It is also known as the community view.

- It is abstract representation of the entire information content of the database.
- It is in general a view of the data as it actually is, that is, it is a 'model' of the 'real world'.
- It consists of multiple occurrences of multiple types of conceptual record, defined in the conceptual schema.
- In order to achieve data independence, the definitions of conceptual records is defined in the conceptual schema.

3. Internal Level

- It is concern with the way the data are physically stored on the hardware.
- Usually the internal level is described using the actual bytes and machine-level terminology which is taken care by the DBMS software.
- The internal view is a low-level representation of the entire database consisting of multiple occurrences of multiple types of internal (stored) records.

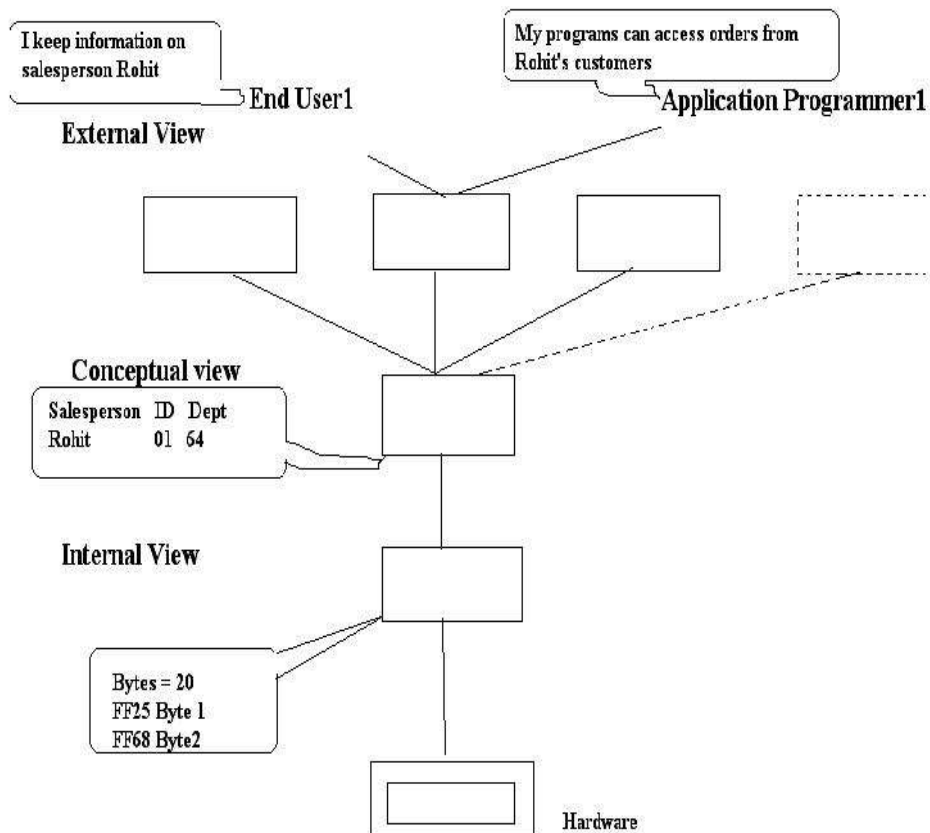


Fig 3.2 Information at different levels of architecture

6

- The above figure represents the different levels of data representation.
- The internal level would describe exactly which bytes contain the information and how it can be accessed.
- If user1 is the payroll clerk, the external view contains the information of salesperson.
- If the application programmer1 is designing billing program, he would need all order information as well as information on the particular sales representative in the external view.
- Consider a possible schema for a student database. The office administrator wants immediate access to student information.
- The records clerk needs to be sure all student fees structure is calculated and stored in database.

Data Independence

- The main advantage of using the data independence is that both the user program and the database can be altered independently of each other.
- Data independence is therefore defined as the capacity to change one level of schema without changing the schema at the next highest level.
- In a conventional system, applications are data dependent which means that the way in which the data is organised in secondary storage and the way in which it is accessed are both dictated by the requirements of the application, and, moreover, that knowledge of the data organisation and access technique is built into the application logic.
- Data independence can be classified into two types

1. Logical data independence

- It is the ability to modify the conceptual schema without affecting the existing external schemas.
- In logical data independence, the users are shielded from changes in the logical structure of the data or changes in the choice of relations to be stored.

- The changes to the conceptual schema, such as the addition and deletion of entities, addition and deletion of attributes, or addition and deletion of relationships must be possible without changing existing external schemas or having to rewrite application programs.
- Only the view definition and the mapping need be changed in a DBMS that supports logical data independence.

2. Physical data independence

- The ability to modify the internal schema without having to change the conceptual or external schemas is called physical data independence.
- In physical data independence, the conceptual schema insulates the users from changes in the physical storage of the data.
- The changes to the internal schema, such as using different file organizations or storage structures, using different storage devices, modifying indexes or hashing algorithms must be possible without changing the conceptual or external schemas.
- In other words, physical data independence indicates that the physical storage structures or devices used for storing the data could be changed without necessitating a change in the conceptual view or any of the external views.
- **Note:** The Logical data independence is difficult to achieve than physical data independence as it requires the flexibility in the design of database and programmer has to anticipate the future requirements or modifications in the design of the database.

3.3 ROLE OF DATABASE USERS

- The database users are classified into four categories.

1. Naive user:

- They are unsophisticated users who interact with the system by invoking one of the permanent application programs that have been written previously.

- Example: Suppose the bank teller wants to transfer the money after maturity of the fixed deposit amount of a particular customer, needs to invoke a program called transfer.
- This program ask the teller for the amount of money to be transferred, the account to which the money is to be transferred.

2. Application programmers

- They are the computer professionals who interact with the system through DML calls, which are embedded in a program written in a host programming language.
- Since the DML syntax is different from the host language syntax, DMI calls are usually prefaced by a special character so that the appropriate code can be generated.
- A special pre-processor, called the DML precompiler, converts the DML statements to normal procedure calls in the host language.
- There are special types of programming languages that combine control structures of Pascal like languages with control structures for the manipulation of a database object.

3. Sophisticated users

- These users interact with the database using database query language.
- They submit their query to the query processor.
- Then Data Manipulation Language (DML) functions are performed on the database to retrieve the data.
- Tools used by these users are OLAP(Online Analytical Processing) and data mining tools.

4. Specialized users

- These users write specialized database applications to retrieve data.
- These applications can be used to retrieve data with complex data types e.g. graphics data and audio data.

3.4 ROLE OF DATABASE ADMINISTRATOR

A person having who has central control over data and programs that access the data is called DBA. Following are the functions of the DBA.

- *Schema definition*: DBA creates database schema by executing Data Definition Language (DDL) statements.
- Storage structure and access method definition
- *Schema and physical organization modification*: If any changes are to be made in the original schema, to fit the need of your organization, then these changes are carried out by the DBA.
- *Granting of authorization for data access*: DBA can decide which parts of data can be accessed by which users. Before any user access the data, DBMS checks which rights are granted to the user by the DBA.
- *Routine maintenance*: DBA has to take periodic backups of the database, ensure that enough disk space is available to store new data, ensure that performance of DBMS is not degraded by any operation carried out by the users.
- *Performance monitoring*: Here DBMS should respond to changes in requirements, i.e. changing details of storage and access thereby organising the system so as to get the performance that is '*best for the enterprise*'.

3.5 TRANSACTION MANAGEMENT

What is a Transaction?

- A transaction is an event which occurs on the database. Generally a transaction reads a value from the database or writes a value to the database.
- Although a transaction can both read and write on the database, there are some fundamental differences between these two classes of operations.
- A read operation does not change the image of the database in any way.
- But a write operation, whether performed with the intention of inserting, updating or deleting data from the database,

changes the image of the database. ie, we may say that these transactions bring the database from an image which existed before the transaction occurred (called the **Before Image** or **BFIM**) to an image which exists after the transaction occurred (called the **After Image** or **AFIM**).

The Four Properties of Transactions

- Every transaction, for whatever purpose it is being used, has the following four properties. Taking the initial letters of these four properties we collectively call them the **ACID Properties**.
- 1. **Atomicity**: This means that either all of the instructions within the transaction will be reflected in the database, or none of them will be reflected.
 - Say for example, we have two accounts A and B, each containing Rs 1000/-.
 - We now start a transaction to deposit Rs 1000/- from account A to Account B.

Read A;

A = A - 100;

Write A;

Read B;

B = B + 100;

Write B;

The transaction has 6 instructions to extract the amount from A and submit it to B. The AFIM will show Rs 90000/- in A and Rs 1100/- in B.

- Now, suppose there is a power failure just after instruction 3 (Write A) has been complete. What happens now? After the system recovers the AFIM will show Rs 900/- in A, but the same Rs 1000/- in B. It would be said that Rs 100/- evaporated in thin air for the power failure. Clearly such a situation is not acceptable.
- The solution is to keep every value calculated by the instruction of the transaction not in any stable storage (hard disc) but in a volatile storage (RAM), until the transaction completes its last instruction.
- When we see that there has not been any error we do something known as a **COMMIT** operation. Its job is to write every temporarily calculated value from the volatile storage on to the stable storage.

- In this way, even if power fails at instruction 3, the post recovery image of the database will show accounts A and B both containing Rs 1000/-, as if the failed transaction had never occurred.
2. **Consistency:** If we execute a particular transaction in isolation or together with other transaction, (i.e. presumably in a multi-programming environment), the transaction will yield the same expected result.
- To give better performance, every database management system supports the execution of multiple transactions at the same time, using CPU Time Sharing.
 - Concurrently executing transactions may have to deal with the problem of sharable resources, i.e. resources that multiple transactions are trying to read/write at the same time.
 - For example, we may have a table or a record on which two transaction are trying to read or write at the same time. Careful mechanisms are created in order to prevent mismanagement of these sharable resources, so that there should not be any change in the way a transaction performs.
 - A transaction which deposits Rs 100/- to account A must deposit the same amount whether it is acting alone or in conjunction with another transaction that may be trying to deposit or withdraw some amount at the same time.
3. **Isolation:** In case multiple transactions are executing concurrently and trying to access a sharable resource at the same time, the system should create an ordering in their execution so that they should not create any anomaly in the value stored at the sharable resource.

There are several ways to achieve this and the most popular one is using some kind of locking mechanism.

- Again, if you have the concept of Operating Systems, then you should remember the semaphores, how it is used by a process to make a resource busy before starting to use it, and how it is used to release the resource after the usage is over.
- Other processes intending to access that same resource must wait during this time. Locking is almost similar. It states that a transaction must first lock the data item that it wishes to access, and release the lock when the accessing is no longer required.

- Once a transaction locks the data item, other transactions wishing to access the same data item must wait until the lock is released.
4. **Durability:** It states that once a transaction has been complete the changes it has made should be permanent.
- As we have seen in the explanation of the Atomicity property, the transaction, if completes successfully, is committed. Once the COMMIT is done, the changes which the transaction has made to the database are immediately written into permanent storage.
 - So, after the transaction has been committed successfully, there is no question of any loss of information even if the power fails. Committing a transaction guarantees that the AFIM has been reached.
 - There are several ways Atomicity and Durability can be implemented. One of them is called **Shadow Copy**.
 - In this scheme a database pointer is used to point to the BFIM of the database. During the transaction, all the temporary changes are recorded into a Shadow Copy, which is an exact copy of the original database plus the changes made by the transaction, which is the AFIM.
 - Now, if the transaction is required to COMMIT, then the database pointer is updated to point to the AFIM copy, and the BFIM copy is discarded.
 - On the other hand, if the transaction is not committed, then the database pointer is not updated. It keeps pointing to the BFIM, and the AFIM is discarded. This is a simple scheme, but takes a lot of memory space and time to implement.

3.6 DATABASE STRUCTURE

In a database structure, the DBMS acts as an interface between the user and the database.

The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database.

The components of DBMS perform these requested operations on the database and provide necessary data to the users.

The various components of DBMS are shown below: -

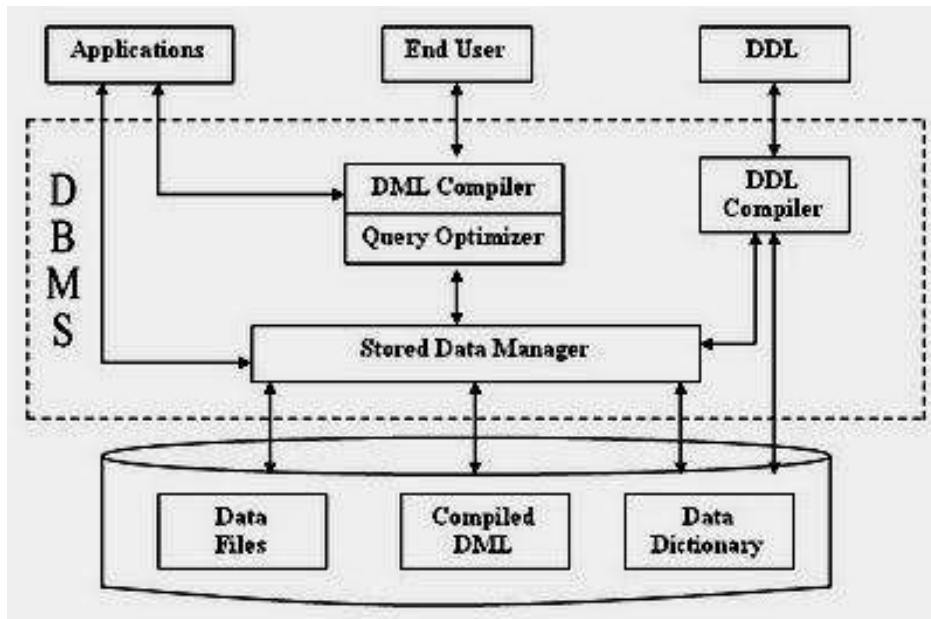


Fig. 3.3 Structure Of DBMS

1. DDL Compiler

Data Description Language compiler processes schema definitions specified in the DDL. It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.

2. DML Compiler and Query optimizer

The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access. The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

3. Data Manager

The Data Manager is the central software component of the DBMS also known as Database Control System.

The Main Functions Of Data Manager are

- It convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.
- It controls DBMS information access that is stored on disk.
- It also controls handling buffers in main memory.

- It also enforces constraints to maintain consistency and integrity of the data.
- It also synchronizes the simultaneous operations performed by the concurrent users.
- It also controls the backup and recovery operations.

4. **Data Dictionary**

Data Dictionary is a repository of description of data in the database. It contains information about

- *Data* - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.
- *Relationships* between database transactions and data items referenced by them which are useful in determining which transactions are affected when certain data definitions are changed.
- *Constraints* on data i.e. range of values permitted.
- Detailed information on physical database design such as storage structure, access paths, files and record sizes.
- *Access Authorization* which is the description of database users their responsibilities and their access rights.
- *Usage statistics* such as frequency of query and transactions.

Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as an important part of the DBMS.

Importance of Data Dictionary

Data Dictionary is necessary in the databases due to following reasons:

- It improves the control of DBA over the information system and user's understanding of use of the system.
- It helps in documentations of the database design process by storing documentation of the result of every design phase and design decisions.
- It helps in searching the views on the database definitions of those views.
- It provides great assistance in producing a report of which data elements (i.e. data values) are used in all the programs.
- It promotes data independence i.e. by addition or modifications of structures in the database application program are not affected.

5. **Data Files** - It contains the data portion of the database.
6. **Compiled DML** - The DML compiler converts the high level Queries into low level file access commands known as compiled DML.
7. **End Users** – They are the users of the system who is going to use the system for their day to day activities.

3.7 SUMMARY

- A database system is a computer-based system to record and maintain information.
- The data dictionary contains logical descriptions of the data and its relationships, physical information about data storage, and usually information on users on users and privileges.
- The three levels of database architecture are
 1. *External Level*: It is concerned with the way individual user observes the data.
 2. *Conceptual Level*: It can be regarded as a community user view a formal description of data of interest to the organisation, independent of any storage considerations.
 3. *Internal Level*: It is concerned with the way in which the data is actually used.
- Data independence is defined as the capacity to change one level of schema without changing the schema at the next highest level.
- Data independence is categorized into two types
 1. Logical data independence- It is the ability to modify the conceptual schema without affecting the existing external schemas.
 2. Physical data independence- It is ability to modify the internal schema without having to change the conceptual or external schemas.
- The different types of database users are naïve users,application programmers,sophisticated users,specialized users.
- A person having who has central control over data and programs that access the data is called Database Administrator who plays the various roles.
- A transaction is an event which occurs on the database.

- The four properties of transactions are generally denoted by ACID.

3.8 MODEL QUESTIONS

1. Explain the structure of database with neat label diagram
2. What is data independence? Why it is needed in database?
3. Explain the different categories of Data independence.
4. What is data abstraction? Explain the different levels of data abstraction.
5. Explain the different types of database users.
6. Explain the different role performed by database administrator.
7. Write a short note on transaction management.



INTRODUCTION TO DATA MODELS

Unit Structure

- 4.0 Objectives
- 4.1 Introduction
- 4.2 Evolution of Data models
- 4.3 Types of Data Models
- 4.4 Merits and Demerits of Each Model
- 4.5 Business Rules
- 4.6 Summary
- 4.7 Review Questions

5.0 OBJECTIVES

4.1 INTRODUCTION

- A *data model* is a picture or description which shows how the data is to be arranged to achieve a given task.
- It is a clear model which specifies how the data items are arranged in a given model.
- Some data models which gives a clear picture which shows the manner in which the data records are connected or related within a file structure. These are called structural data models.
- DBMS organize and structure data so that it can be retrieved and manipulated by different users and application programs.

- The data structures and access techniques provided by a particular DBMS are called its data model.
- A data model determined both the personality of a DBMS and the applications for which it is particularly well suited.

4.2 EVOLUTION OF DATA MODELS

- The first non-proprietary programming language was COBOL and with COBOL, and later FORTRAN, programming became the foundation of creating enterprise computer systems.
- The systems developed, needed to store its data somewhere and the programmers designed more or less proprietary and specialized solutions for this purpose.
- In 1964 the first commercial database management system (DBMS) was born; IDS - Integrated Data Store, developed at General Electric, based upon an early network data model developed by C.W Bachman (Bachman 1965).
- In the late 1960s, IBM and North American Aviation (later Rockwell International) developed IMS - Information Management System, and its DL/1-language. This was the first commercial hierarchical DBMS. Both kinds of DBMSs (hierarchical and network) were accessible from the programming language (usually COBOL) using a low-level interface. This made the task of creating an application, maintaining the database as well as tuning and development controllable, but still complex and time-consuming.
- In 1970 Edgar F. Codd published an article which offered a fundamentally different approach (Codd 1970).
- Codd suggested that all data in a database could be represented as a tabular structure (tables with columns and rows, which he called relations) and that these relations could be accessed using a high-level non-procedural (or declarative) language.
- Instead of writing algorithms to access data, this approach only needed a predicate that identified the desired records or combination of records. This would lead to higher programmer productivity and in the beginning of the 1980s several Relational DBMS (RDBMS) products emerged (e.g. Oracle, Informix, Ingres and DB2).

- As the DBMSs evolved, so did the programming languages. In 1967 Simula, the first object-oriented programming language was born. Simula was developed to make a foundation to develop simulation programs, and contained the now familiar class-concept. Several other programming languages adopted the class-concept from Simula (e.g. C++, Java, Eiffel, and Smalltalk) and continued to evolve more or less independently of the DBMSs.
- In the early 1980s research started on another kind of database. This research was among other things, motivated by the need of a database system capable of handling complex objects and structures like those used in CAD systems, CASE and OIS systems (Zdonik. 1994). To accomplish these tasks the database had to be able to store classes and objects and the objects associations and methods, and the object-oriented DBMS (OODBMS) emerged. In the late 1980s several vendors had developed OODBMSs (e.g. ObjectDesign, Versant, O2 and Objectivity).
- OODBMSs were no threat in the late 1980s to the now big commercial vendors developing and selling hierarchical, network or relational databases. In 1991 ODMG (Object Database Management Group) was founded, mainly thanks to Rick Cattell of JavaSoft, and in 1993 several vendors of OODBMSs agreed upon an OODBMS standard called ODMG-93.
- The relational databases already had its standard - SQL-92, defined by its ANSI committee and ISO. And so did the network database vendors as well; CODASYL (defined in 1986 by the ANSI X3H2 committee).
- The founding of ODMG and the fact that object-oriented programming languages became more and more used may well have been the major driving forces when the ANSI X3H2 committee started its work on SQL3 in 1992. This proposal put another type of DBMS on the arena - the object relational DBMS (ORDBMS).
- While all this was happening, more and more programmers converted from C and other languages to C++. C++ was becoming the most used object-oriented language, but C++ application was not always that easy to develop and maintain. Such applications often had memory-leaks, erroneous pointers and other trivial problems attached to them.

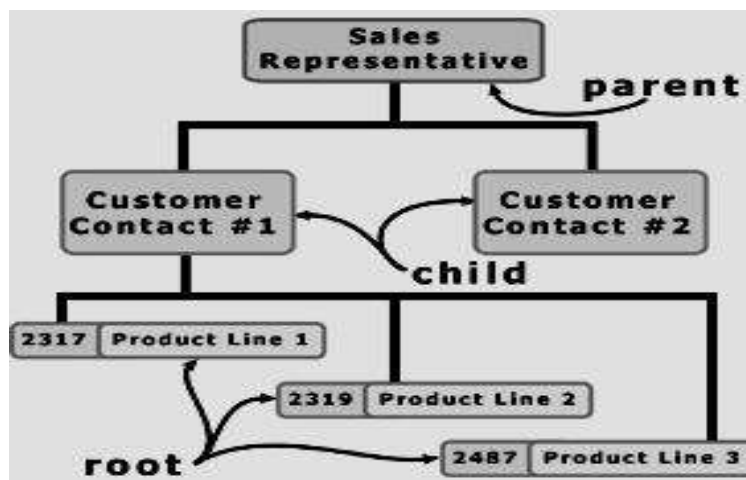
- In 1991 Sun's Green Team started the development of a new programming language which was loosely based on C++. The language was named Oak after the trees outside the office window of the language designer - James Gosling.
- In 1992 Sun turned Green Team into a fully owned company, called First Person Inc. National Center for Supercomputing introduced Mosaic in 1993, a WWW browser, and the Internet began to bustle with traffic. Soon other WWW browser followed.
- In 1994 First Person built an Oak-ready browser called WebRunner and Sun backed the decision to give the language (Oak) away for free, but first Oak was renamed to Java and WebRunner to HotJava. Java became available to millions of people due to Netscapes bundling of Java, and soon others followed (Bank 1995).

4.3 TYPES OF DATA MODEL

There are four different types of data models

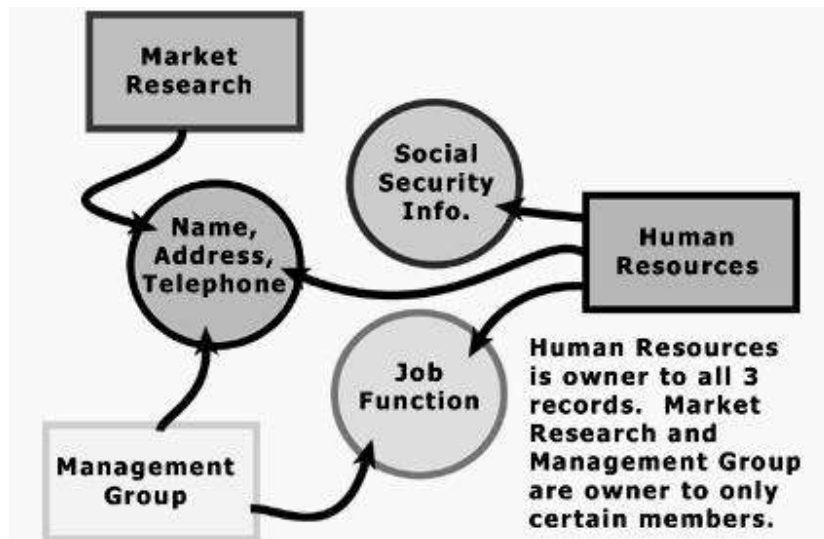
- 4.3.1 Hierarchical databases
- 4.3.2 Network databases
- 4.3.3 Relational databases
- 4.3.4 Object oriented databases

4.3.1 Hierarchical databases



- **Hierarchical Databases** is most commonly used with mainframe systems.
- It is one of the oldest methods of organizing and storing data and it is still used by some organizations for making travel reservations.
- A hierarchical database is organized in pyramid fashion, like the branches of a tree extending downwards.
- In this model, related fields or records are grouped together so that there are higher-level records and lower-level records, just like the parents in a family tree sit above the subordinated children.
- Based on this analogy, the parent record at the top of the pyramid is called the **root record**.
- A child record always has only one parent record to which it is linked, just like in a normal family tree.
- In contrast, a parent record may have more than one child record linked to it. Hierarchical databases work by moving from the top down.
- A record search is conducted by starting at the top of the pyramid and working down through the tree from parent to child until the appropriate child record is found. Furthermore, each child can also be a parent with children underneath it.

4.3.2 Network databases



- **Network databases** are similar to hierarchical databases by also having a hierarchical structure. There are a few key differences, however.

- Instead of looking like an upside-down tree, a network database looks more like a cobweb or interconnected network of records. In network databases, children are called **members** and parents are called **owners**.

The most important difference is that each child or member can have more than one parent (or owner).

- Similar to hierarchical databases, network databases are principally used on mainframe computers.
- Since more connections can be made between different types of data, network databases are considered more flexible. However, two limitations must be considered when using this kind of database.
- Similar to hierarchical databases, network databases must be defined in advance. There is also a limit to the number of connections that can be made between records.

4.3.3 Relational databases

- Pre-relational models depended upon being able to determine explicitly where and how individual records were stored.
- Early relational proponents argued that the relational data model viewed information *logically* rather than *physically*, but this is not quite correct.
- Earlier data models associated the logical and physical aspects of information together; logically-related information was stored in physical proximity within a data file. The relational data model first separated the logical from the physical aspects.
- The relational data model looks at information as an *unordered* collection of "relations."
- Each relation is populated with *unordered* "tuples" of the same *unordered* "field" structure.
- Fields may only contain values of a well-defined ("atomic") domain or the *null* value. The unordered aspect needs to be emphasized. For expository purposes, relations are often viewed as "tables".
- The tuples constitute the "rows" of the table; values for a specific field constitute "columns". However, the "table data model" tends to impose a very non-relational ordering on both

tuples and fields. Relations are an abstraction of how data is stored; tables are just one of many possible implementations.

- Some of the relational terms are crafted to emphasize the distinction between logical and physical features, to avoid confusing one concept with another. However, vocabulary leakage from other disciplines has sprinkled into the conversation of relational proponents.
- There is a strong tendency to refer to an individual tuple/row as a "record" because collections of fields in other models are called records. "Attribute" is often used synonymously with field.
- To be sure, "unordered" implies neither "chaotic" nor "random". Relations and Fields are named uniquely and identified easily. Distinguishing between tuples is more subtle since the order is not pre-defined.
- Rather than depending upon relative (as in hierarchy) or absolute (as in network) locations, tuples may only be differentiated according to their contents.
- Consequently, duplicate tuples are not permitted within a single relation. Even more strongly, distinct tuples must have a unique "key" (some combination of a relation's named fields).
- The set of minimal keys includes one "primary key"; the rest are "candidate keys". Within a tuple, references to other tuples are expressed as a "foreign key," which should contain the values of the referenced tuple's primary key.
- Relational theory provides a firm mathematical foundation for data management. Set theory could be applied to relations using relational algebraic operations (union, intersection, join, projection, etc.).
- Assertions about the existence or non-existence of some condition with a data base could be proven with a rigor unachievable with earlier models.

4.3.4 Object oriented databases

- A data model is a logic organization of the real world objects (entities), constraints on them, and the relationships among objects. A DB language is a concrete syntax for a data model. A DB system implements a data model.

- A core object-oriented data model consists of the following basic object-oriented concepts:

- (1) **object and object identifier:** Any real world entity is uniformly modeled as an object (associated with a unique id: used to pinpoint an object to retrieve).
- (2) **attributes and methods:** Here every object has a state (the set of values for the attributes of the object) and a behavior (the set of methods - program code - which operate on the state of the object). The state and behavior encapsulated in an object are accessed or invoked from outside the object only through explicit message passing.

An attribute is an instance variable, whose domain may be any class: user-defined or primitive. A class composition hierarchy (aggregation relationship) is orthogonal to the concept of a class hierarchy. The link in a class composition hierarchy may form cycles.

- (3) **class:** a means of grouping all the objects which share the same set of attributes and methods. An object must belong to only one class as an instance of that class (instance-of relationship). A class is similar to an abstract data type. A class may also be primitive (no attributes), e.g., integer, string, Boolean.
- (4) **Class hierarchy and inheritance:** derive a new class (subclass) from an existing class (superclass). The subclass inherits all the attributes and methods of the existing class and may have additional attributes and methods. single inheritance (class hierarchy) vs. multiple inheritance (class lattice).

4.4 ADVANTAGES AND DISADVANTAGES OF DATA MODELS

Advantages

1. **Simplicity:** Since the database is based on the hierarchical structure, the relationship between the various layers is logically simple.
2. **Data Security:** Hierarchical model was the first database model that offered the data security that is provided by the DBMS.
3. **Data Integrity:** Since it is based on the parent child relationship, there is always a link between the parent segment and the child segment under it.

4. **Efficiency:** It is very efficient because when the database contains a large number of 1:N relationship and when the user require large number of transaction.

Disadvantages

1. **Implementation complexity:** Although it is simple and easy to design, it is quite complex to implement.

2. **Database Management Problem:** If you make any changes in the database structure, then you need to make changes in the entire application program that access the database.

3. **Lack of Structural Independence:** there is lack of structural independence because when we change the structure then it becomes compulsory to change the application too.

4. **Operational Anomalies:** Hierarchical model suffers from the insert, delete and update anomalies, also retrieval operation is difficult.

4.4.2 Network Model

Advantages

1. **Conceptual Simplicity:** just like hierarchical model it also simple and easy to implement.

2. **Capability to handle more relationship types:** the network model can handle one to one 1:1 and many to many N: N relationship.

3. **Ease to access data:** the data access is easier than the hierarchical model.

4. **Data Integrity:** Since it is based on the parent child relationship, there is always a link between the parent segment and the child segment under it.

5. **Data Independence:** The network model is better than hierarchical model in case of data independence.

Disadvantages

1. **System Complexity:** All the records have to maintain using pointers thus the database structure becomes more complex.

2. Operational Anomalies: As discussed earlier in network model large number of pointers is required so insertion, deletion and updating more complex.

3. Absence of structural Independence: there is lack of structural independence because when we change the structure then it becomes compulsory to change the application too.

4.4.3 Relational Model

Advantages

1. Conceptual Simplicity: We have seen that both the hierarchical and network models are conceptually simple, but relational model is simpler than both of those two.

2. Structural Independence: In the Relational model, changes in the structure do not affect the data access.

3. Design Implementation: the relational model achieves both data independence and structural independence.

4. Ad hoc query capability: the presence of very powerful, flexible and easy to use capability is one of the main reason for the immense popularity of the relational database model.

Disadvantages

1. Hardware overheads: The relational database systems hide the implementation complexities and the physical data storage details from the user. For doing this, the relational database system need more powerful hardware computers and data storage devices.

2. Ease of design can lead to bad design: The relational database is easy to design and use. The user needs not to know the complexities of the data storage. This ease of design and use can lead to the development and implementation of the very poorly designed database management system.

4.5 BUSINESS RULES

- Business rules are the rules that are created to affect the way your business works. Usually, these are rules that involve employees or staff and are rules that specify what they can and cannot do.
- A great example of a business rule involves marriages. For many companies, a boss is not allowed to marry an employee or an accountant at a company is usually not allowed to marry another accountant.

- In this case, the accountants are not allowed to be married because there is a more likely chance that the spouses can change financial information and then cover for one another.
- These rules are intended to prevent disruption in a company or business.
- Business Rules are used every day to define entities, attributes, relationships and constraints.
- Usually though they are used for the organization that stores or uses data to be an explanation of a policy, procedure, or principle.
- The data can be considered significant only after business rules are defined, without them it's just records, but to a business they are the characteristics that are defined and seen by the company.
- Business Rules help employees focus on and implement the actions within the organizations environment.
- Some things to think about when creating business rules are to keep them simple, easy to understand, keep them broad so that everyone can have a similar interpretation. To be considered true, business rules must be in writing and kept up to date.
- Identifying business rules are very important to the database design. Business rules allow the creator to develop relationship participation rules and constraints and to create a correct data model.
- They also allow the creators to understand business processes, and the nature, role and scope of the data.
- They are a communication tool between users and creators, and they also help standardize the company's view of the data.
- It is important to keep in mind that some business rules cannot be modeled.
- Business Rules give the proper classification of entities, attributes, relationships, and constraints.
- Sources of business rules are managers, policy makers, department managers, written documentation, procedures, standards, operation manuals, and interviews with end users.

Some examples of business rules:

Departments-----offers-----Course
 Course-----generates-----Class
 Professor -----teaches-----Class

- There are several protocols to the way business rules are written. Not every protocol has to be followed, but in general, a

well-written set of business rules consist of having a unique identifier, describes one and only one concept, are written in plain language, are written, and are from a single source.

- In terms of a unique identifier, business rules should come with an identifier that may consist of the rule number and the department it affects. An example would be 'BRacc01'. In this case, this business rule (BR) is directly related to the accounting department.
- Another important aspect of business rules consist of how the rules are shared within the company.
- A protocol for business rules that many follow is that the business rules are written down. However, with many businesses sharing information directly over the internet, some are opting to place their business rules online in company blogs, wikis, and websites.
- This shares the business rules with all employees faster and easier. In relation to how business rules are shared, it is very important that business rules are written in plain language.
- If business rules are written at a high level language, there is an increased chance that not every person will understand what the business rules cover or what is acceptable and what is not.

4.6 SUMMARY

- A *data model* is a picture or description which shows how the data is to be arranged to achieve a given task.
- The data structures and access techniques provided by a particular DBMS are called its data model.
- In 1964 the first commercial database management system (DBMS) was developed widely known as Integrated Data Store (IDS).
- A hierarchical database is organized in pyramid fashion, like the branches of a tree extending downwards.
- In hierarchical model, the parent record at the top of the pyramid is called the **root record** and the leaf node is called the **child record**.
- **Network databases** are similar to hierarchical databases by also having a hierarchical structure.
- The relational model organizes the records and stores the records in rows and columns.

4.7 REVIEW QUESTIONS

- 1) Explain the need for the data model.
- 2) Write in detail about the history of data model.
- 3) Write a short notes on
 - a. Hierarchical Model
 - b. Network Model
 - c. Object Oriented Model
 - d. Relational Model
- 4) Explain the merit and demerits of hierarchical model.
- 5) Explain the merit and demerits of network model
- 6) Explain the merit and demerits of Relational model.



Unit Structure

5.0 Objectives

5.1 Database design

5.2 ER-Model

5.3ER Diagram

5.4 Constraints on relationship

5.5 Relational Schemas

5.0 OBJECTIVES

The database design process consists of a number of steps listed below. We will focus mainly on step 2, the conceptual database design, and the models used during this step.

Step 1: Requirements Collection and Analysis

- Prospective users are interviewed to understand and document data requirements
- This step results in a concise set of user requirements, which should be detailed and complete.
- The functional requirements should be specified, as well as the data requirements. Functional requirements consist of user operations that will be applied to the database, including retrievals and updates.

- Functional requirements can be documented using diagrams such as sequence diagrams, data flow diagrams, scenarios, etc.

Step 2: Conceptual Design

- Once the requirements are collected and analyzed, the designers go about creating the conceptual schema.
- Conceptual schema: concise description of data requirements of the users, and includes a detailed description of the entity types, relationships and constraints.
- The concepts do not include implementation details; therefore the end users easily understand them, and they can be used as a communication tool.
- The conceptual schema is used to ensure all user requirements are met, and they do not conflict.

Step 3: Database Implementation

- Many DBMS systems use an implementation data model, so the conceptual schema is transformed from the high-level data model into the implementation data model.
- This step is called logical design or data model mapping, which results in the implementation data model of the DBMS.

Step 4: Physical Design

- Internal storage structures, indexes, access paths and file organizations are specified.
Application programs are designed and implemented

ER Model

In software Engineering, an entity relational model is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called **entity-relationship diagrams, ER diagrams, or ERDs**.

In 1976, Entity relationship model developed by Chen,

ER Model is high level Conceptual model which used Conceptual design of database whereas relational model are used to logical design of database

ER Diagram

- A database can be modeled as

A collection of entities

Relationship among the entities

- An entity is a real world object that exists and it is distinguishable from other entities

Example: Person, company, event, plant

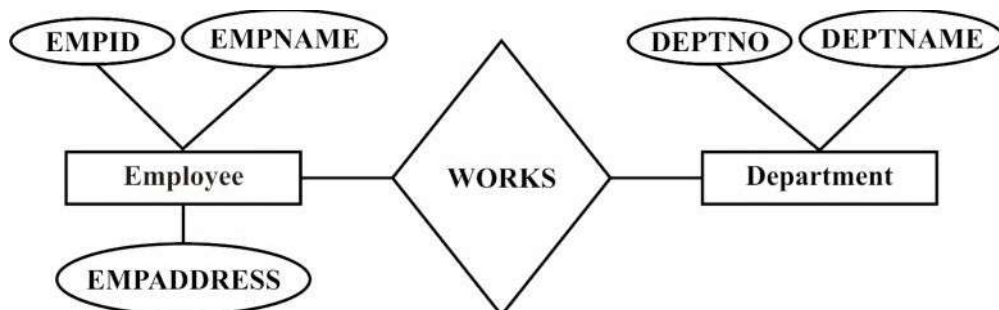
- All the entities in the data model have attributes as known as properties of an entity

Example: people have names and addresses

An Entity set is a set of entities of all same type that share the same properties.

Example: set of all persons, companies, trees, holidays

ER Diagram



- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Underline indicates primary key attributes
- Ellipses represent an attributes
- Double Lines represent total participation of an entity in a relationship set
- Double rectangle represent a weak entity sets

Strong Entity type

An entity type which has own distinct primary key that used to identify specific uniquely from another entity type is called as Strong Entity type

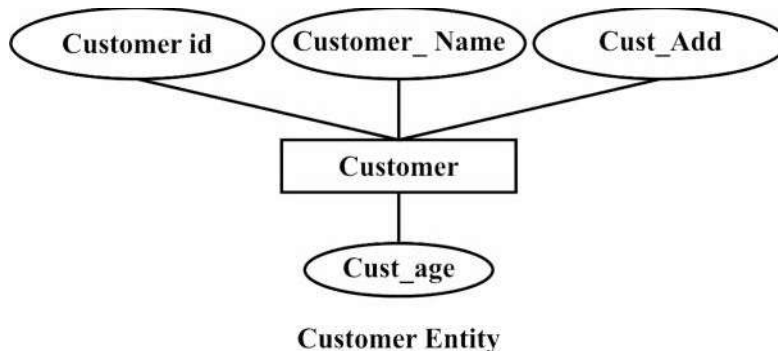
An Entity type which is independent on some other entity type icalled Strong Entity type

Example

In the Case of Client entity Client_no is the primary key of Client entity which is used to uniquely identified among the Client 's entity set

In the case of Customer Entity , Customer_id is the primary key of Customer Entity which is used to uniquely identified among the Customer's entity set

Strong Entity type is represented by rectangle Symbol



Weak entity Type

Entity type which is dependent on some other entity type is called as Weak entity type

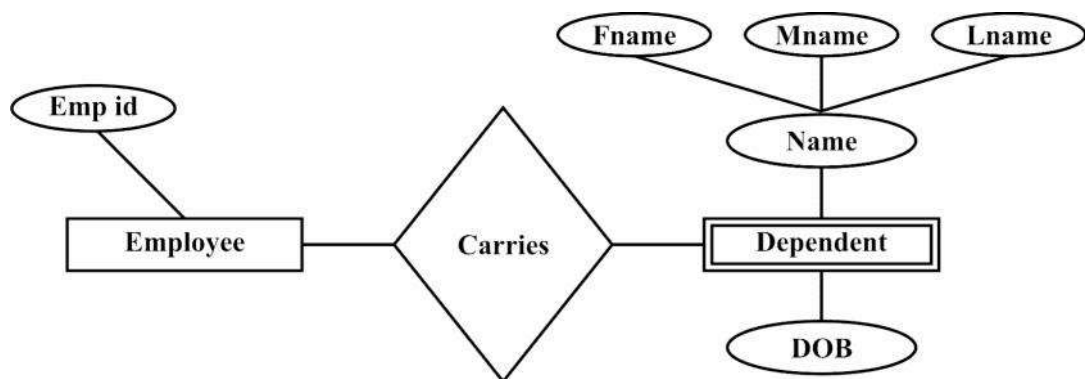
- Weak entity type is dependent on a strong entity and cannot exist on its own
- It does not have a unique identifier that has partial identifier
- Partial identifier is represented by double-line

Some weak entities assign partial identifiers and such partial identifiers of an weak entity called as discriminator

Weak entity type is represented by double rectangle.

Identify relationship

Strong entity type is link with the weak entity type



Dependent entity depend upon Employee entity for primary key

Attributes

Properties of an entity or relationship type is called as attribute Example Staffno, staffname,staff_designation is describes an entity Staff Value of an attribute play a major role of data stored in database Each entity will have the value which is assigned to its attributes Consider an example

Above stated example of Staff Entity which has the attribute named as staffno, the value which is assigned to the staff attribute is '101' and the staffname attribute has the value is 'Mahendra, and staff_designation attribute has the value is 'Manager'

Attribute domains

The set of allowable values which is assigned to one or more attribute is knowns as Attribute domains

There are types of attributes has been classified Such as simple and Composite type,single valued and multi valued attributes Stored and derived attributes, null attributes and Key attributes

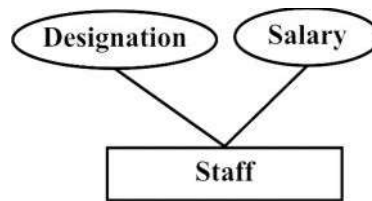
1) Simple Attributes

Simple attributes is an attributes which can further divided in to two parts

Or

An Attribute composed of single component with an independent existence

For an example: Designation of an staff and Salary of an staff



Simple Attributes

Composite Attribute

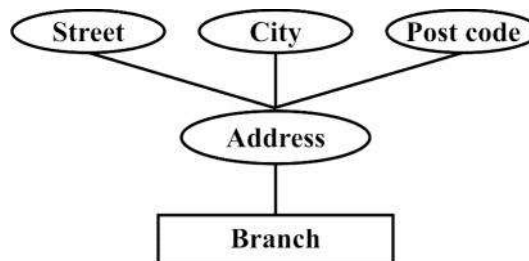
Composite Attribute is an attribute which is futher divided into many parts

Or

An attributed composed of multiple component, each component has its own independent existence

Example

Address attributes of an Branch entity that can be further divided in to sub parts i.e street, city and postalcode as an attributes



Composite Attributes

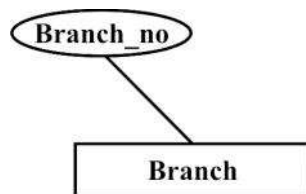
2) Single valued and Mutli Valued attributes

Single valued attribute is an attribute which as single value(atomic) for each entity.

Or

An attribute that holds a single valuefor each occurrence of an entity type

Example: Each branch has only single valued attributes is known as branch_no



Single Valued attributes

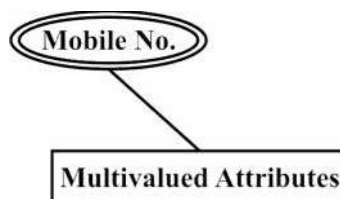
Mutli valued attributes

Mutli valued attribute is an attributes which as many values for each entity

Or

An attribute that holds multiple values for each occurrence of an entity type.

Example : Each staff member has multiple mobile numbers



Multivalued Attributes

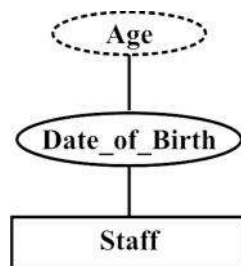
3) Stored and Derived attributes

Stored attributes is an attribute which is used supplied a value to the related attribute

Example Date_of_Birth of an staff is a stored attributes

Derived attributes

The value from the derived attribute is derived from the stored attribute for an example Date_of_Birth is a stored attribute for an each staff member . The value for an Age can be derived from the Date_of_Birth attributes I.e by subtracting the Date_of_Birth from the Current date, therefore the Stored attributes is used supplied a value to the related attributes



Null attribute

The attribute which take NULL value when entity does not have the value to it.

The Null attribute is an attribute their value is unknown, unassigned and missing information

Key Attributes

This attribute has the unique value for an entity which is used to identified given row in the table is called as key attribute of an entity

Example : Staff_no is an key attribute which has an unique value which is used to identifies given row in the table

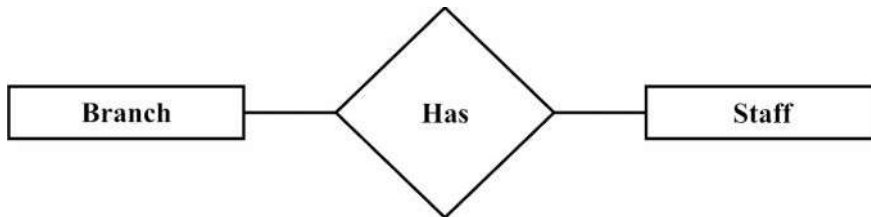


Relationships

A set of meaningful relationship among several entities

We used to indicate the diamond symbol for Relationships among the several entities, it could read from left to right

Example : Branch has a staff



Degree of relationship

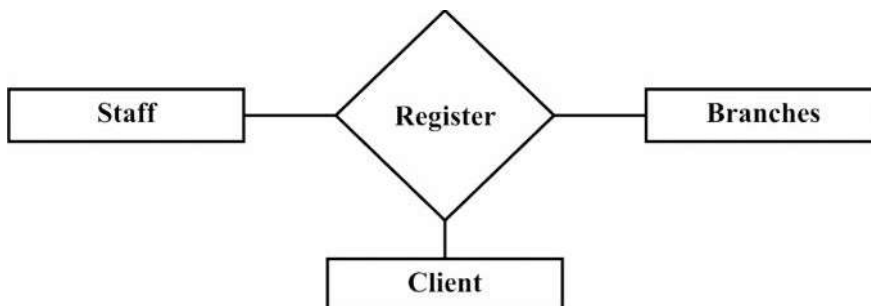
It is the number of entities participated in a particular relational model

There are two type of degree of relationship.

Binary relationship: A Relationship of degree two is called as binary relationship

Ternary Relationship: A relationship of degree three is called as Ternary relationship.

Example



Staff registers a Client at a branch

Relationship set

The collection of similar relationship is known as Relationship set

Constraints on relationship

1) Mapping Constraints / Cardinalities

The number (or range) of possible entity type that is associated to another entity type through a particular entity

Cardinalities indicates that a specific number of entity occurrence of related entity .

Type of Mapping Constraints

One-to-one (1:1)

One-to-many (1:*)

Many- to-one(*:1)

Many-to-many (*:*)

Type **One-to-one (1:1)**

In this type of Mapping Constraint One record of an entity is related to the one record of an another entity

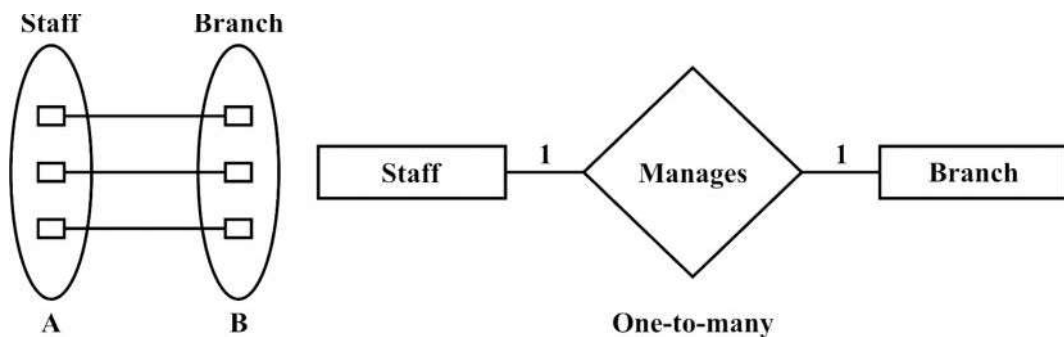
That is one row on an table is related to an one row of another table

i.e A is associated with at most one entity in B and B is associated with at most one entity in A

Example

Each branch is managed by one member of the staff that's means Branch Manager

A member of staff can manage zero or one branch



2) One- to- many

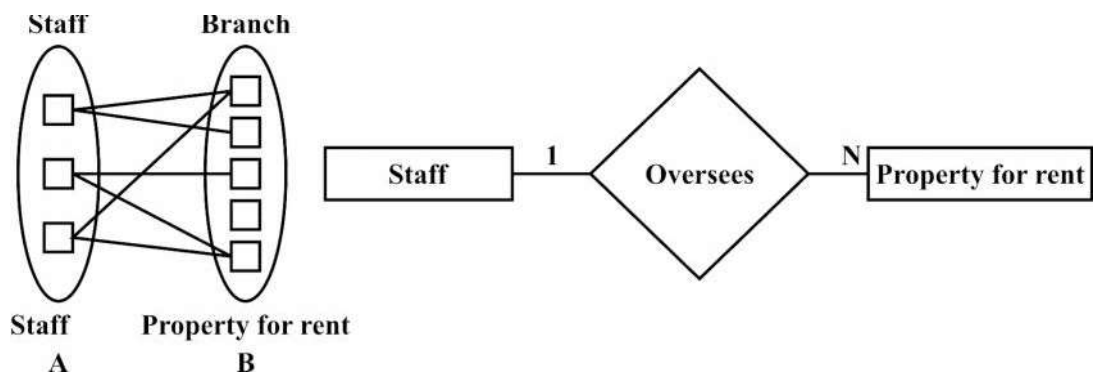
In this constraints, One record in the entity can be related with many record in other entity

A is associated with any number of entities in B

B is associated with at most one entity in A

E.g. each member of staff oversees zero or more prosperity for rent

Every row in the Staff table can have relationship with many rows in the propertyforRent Table

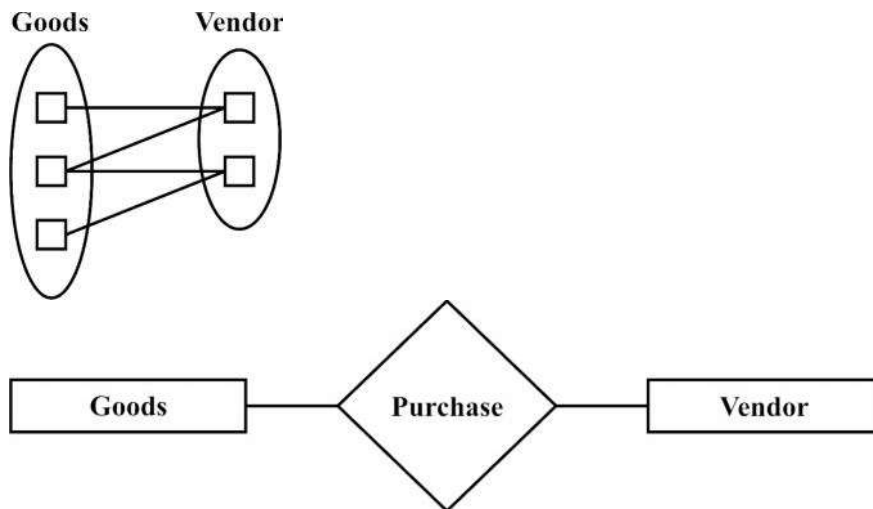


One To Many

In this type Mapping Constraints , Many records in the one entity is related to the only one records in the other entity

An entity in A is associated with at least one entity in B . an entity in B can be associated with any number of entities in A.

Example one vendors has many Goods and Many Goods is purchase by one Vendors

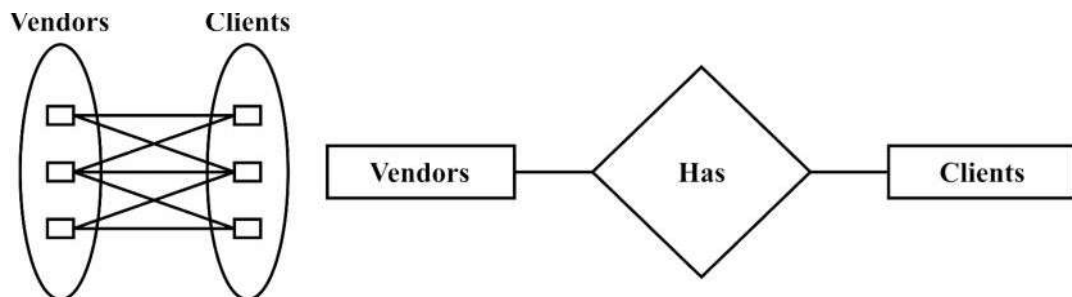


Many to Many

In this Mapping Constraints , Many records in the entity is related Many records in the other entity

An entity in A is associated with any number of entities in B. and an entity in B is associated with any number of entities in A.

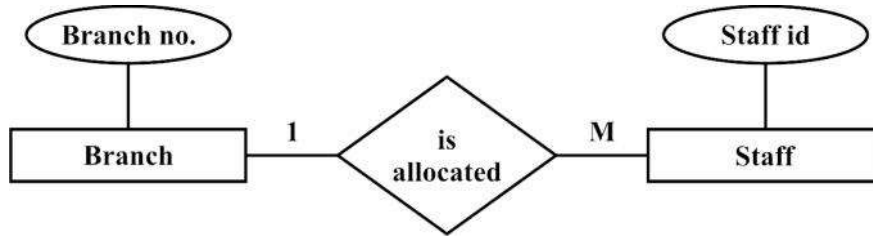
Many Vendors Has Clients and Many Clients has may Vendors



Participation Constraints

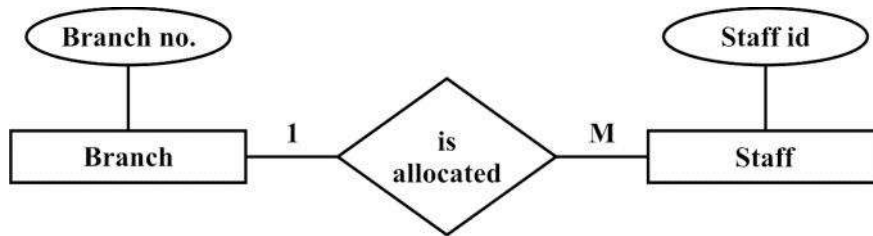
There are two types of participation constraints:
 Total Participation: Every Instance of the first Entity type must share with on or more instances of the relationship type with the other entity type.

The total participation is represented by a dark line or double line between the relationship and entity



Every Branch office is allocated members of Staff

Partial Participation: There exist an instance of the first entity type that don't share an instance of the relationship type with the other entity type.



A member of Staff need not work at a Branch office

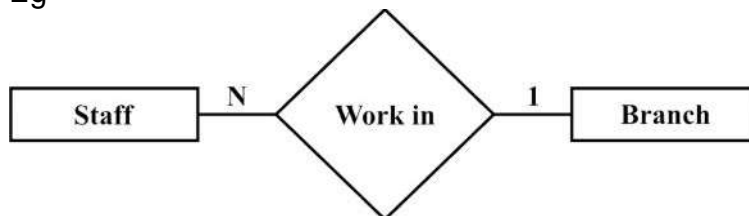
Notations used In ER Diagrams For Representing Relations

1) Cardinality Ratio Notation

In this method ,Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N

Shown by placing appropriate numbers on the relationship edges

Eg

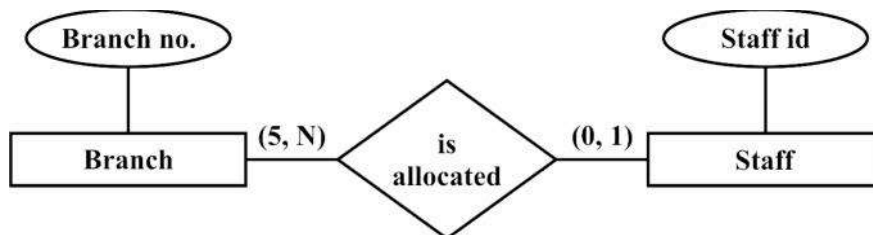


Number of Staffs working in Branch

2) Min –Max notation

The alternate of notation by specify the pair of integer, that used to specify the minimum and maximum participation of each entity type in the form of (min, max)

The Minimum participation of 0 indicate partial participation where as maximum participation of 1 or more indicates total participation



At least 5 staff is allocated to branch

Limitation Of Entity Relationship Model

Problems may arise when designing a conceptual data model called connection traps.

- Often due to a misinterpretation of the meaning of certain relationships.

Extended Entity Relation Relationship Model

Since 1980 there has been increase in the emergence of new database application with more demanding application

Basic concepts of ER modeling are not sufficient to represent the requirement of newer, more complex operation

To overcome the issue of ER modeling there is response in development of additional 'semantic' modeling concept

Semantic concept which is integrated into original ER Model is known as Extended Entity Relation Relationship Model (EER)

Additional Concept which is includes in the Extended Entity Relation Relationship Model are specialization/ generalization, categorization, superclass/subclass, attribute inheritance

Extended EER Model is used the concept of object oriented such as inheritance

Sub classes and Super classes

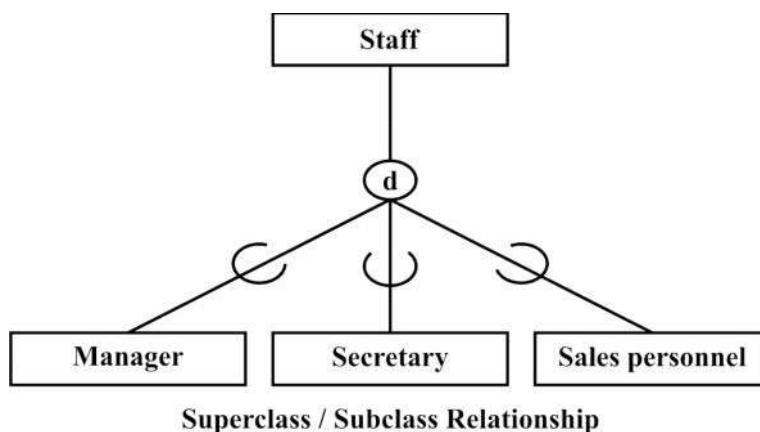
In some case , entity type has numerous sub-grouping of its entities because that are meaningful way for representation and need to be explicitly defined because of their importance

The set listed is a subset of the entities that belong to the staff entity which means that every entity that belongs to one of the sub sets is also an Staff

An entity type that includes distinct Subclasses that require to be represented in a data model is called as super class.

A Subclass is an entity type that has a distinct role and is also a member of the Superclass.

Staff is the super class where as manager, Secretary, sales personnel is the subclass



Superclass /Subclass Relationship

Superclass /Subclass Relationship

The relationship between super class and subclass is called Superclass /Subclass Relationship

In Superclass /Subclass Relationship, the encircled 'd' Indicates that there is Superclass /Subclass Relationship, it is denoted by the symbol

Hence Superclass /Subclass Relationship lead to the object oriented Concept is called as Inheritance

As the above diagram, Arc drawn above the line towards Subclass indicated inheritance Relationship

Type Inheritance

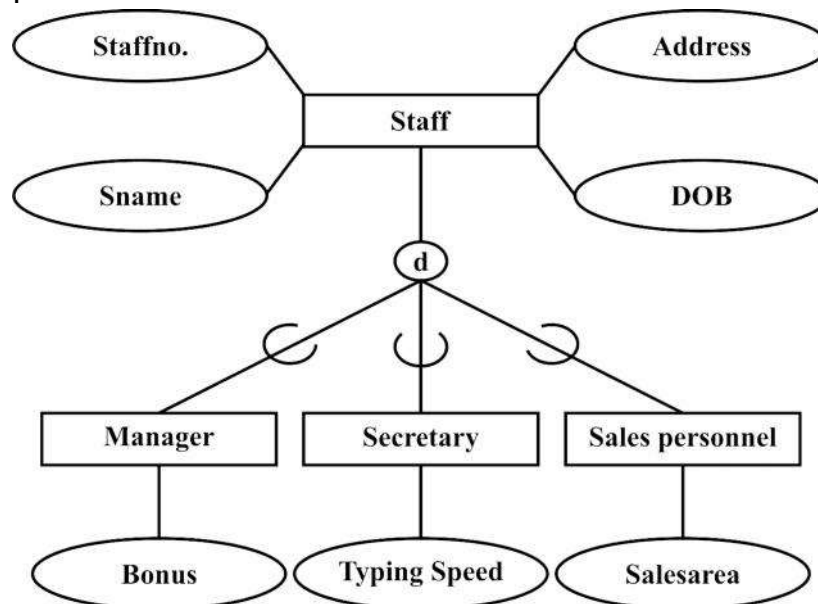
- The type of an entity is defined by the attributes it possesses, and the relationship types it participates in.
- Because an entity in a subclass represents the same entity from the super class, it should possess all the values for its attributes, as well as the attributes as a member of the super class.
- This means that an entity that is a member of a subclass inherits all the attributes of the entity as a member of the super class; as well, an entity inherits all the relationships in which the super class participates.

Specialization

The process of defining a set of subclasses of super class

The specialization is a top down approach of super class and subclasses

The set of sub classes is based on some distinguishing characteristic of the super class.



Notation for Specialization

- To represent a specialization, the subclasses that define a specialization are attached by lines to a circle that represents the specialization, and is connected to the super class.

- The subset symbol (half-circle) is shown on each line connecting a subclass to a super class, indicates the direction of the super class/subclass relationship.
- Attributes that only apply to the sub class are attached to the rectangle representing the subclass. They are called specific attributes.

A sub class can also participate in specific relationship types

Reasons for Specialization

- Certain attributes may apply to some but not all entities of a super class. A subclass is defined in order to group the entities to which the attributes apply.
- The second reason for using subclasses is that some relationship types may be participated in only by entities that are members of the subclass.

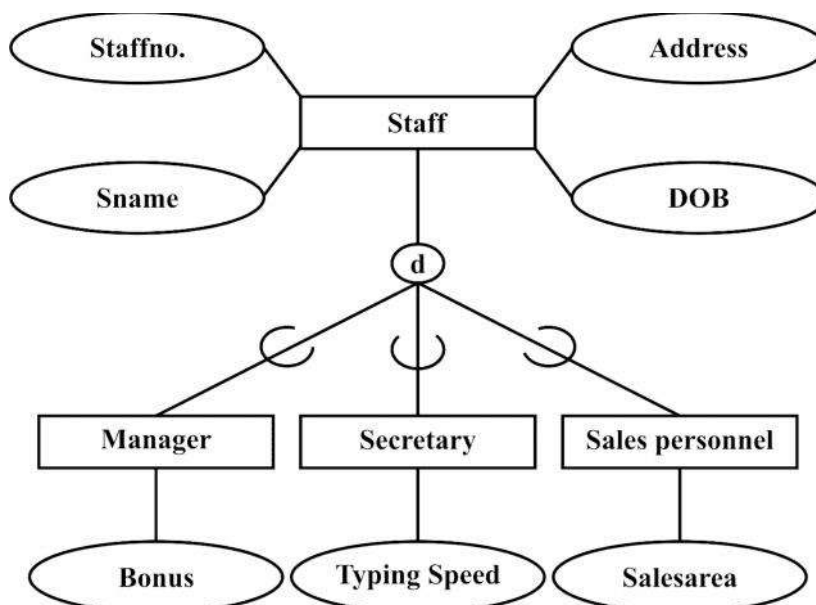
Summary of Specialization

Allows for:

- Defining set of subclasses of entity type
- Create additional specific attributes for each sub class
- Create additional specific relationship types between each sub class and other entity types or other subclasses.

Generalization

- Generalization is the reverse of specialization and this is a bottom-up approach
In Generalization, there are Several classes with common features and generalizing into a super class.



Attribute Inheritance

- An entity in a Subclass may possess subclass specific attributes, as well as those associated with the Superclass

5.6 CODD'S RULE

Dr. E.F Codd was inventor of the relational database model. This model say that whether the Database management system follow the relational model or not and what extents model is relational.

The article mentioned by Dr.E.F.Codd that according to these rule, There is no database management system fully implements all the 12 rules what he has been specified

In 1990, The codd rules extended 12 to 18 rules that's includes catlog,datatypes,authorization etc.

OverView of codd's rule

Sr.NO	Rule	Description
1	The information rule:	All the information in the database should be represented in the term of relational or table.Information should be stored as an values in a tables
2	The guaranteed access rule	All data must be accessible. The Rule say that there is fundamental requirement of primay key for each record in table ,and there should be no ambiguity by stating the table name and its primary key of the each record in the table along with columns name to be accessed
3	Systematic treatment of null values:	Null values could not be treated as blank space or zero values, The null values are known as unknown values, unassigned values should be treated as missing information and inapplicable information that should be treated as systematic , distinct from regular values

4	Active online catalog based on the relational model:	The system must support an online catalog based data dictionary which hold the information or description about the table in the database
5	The comprehensive data sublanguage rule:	<p>The system must support at least one relational language that through which the data in the database must be accessed</p> <p>1 The language can be used both interactively and within application programs.</p> <p>The Language must Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).</p>
6	The view updating rule:	All the view must be theoretically updatable can be updated by the system
7	High-level insert, update, and delete:	This rules states that in the relational model, the structured query language must performed data manipulation such as inserting ,updating and deleting record on sets of rows in the table
8	Physical data independence:	Any change made in the data is physically stored in term of data is stored in the file system through array and link list must not effect application that access the data structure

9	Logical data independence:	This rule state that changes in the logical level(rows ,columns and so on) must not change to the application's structure
10	Integrity independence:	Data integrity constraints should be considered as separated from application program, the structured query language which defines data integrity constraint must be stored in the database in term of data in table that is, in the catalog and not in the application.
11	Distribution independence:	The rules states that the data can be stored centrally on the single machine or it can be stored in the various location(ditributed) on various machine but it should be invisible to the user i.e the user does not location of data is stored whether on the single machine(Centrally stored) or the distributed stored.If the location of database in change then the existing application must continually access the change database
12	The nonsubversion rule:	The system must not have features that allow you to subvert database structure integrity. Basically, the system must not include back doors that let you cheat the system for features such as administrative privileges or data constraints.

Codd's rule in detail

1) The information rule:

- I) All the information in the database should be represented in the term of relational or table. Information should be stored as an values in a tables
- II) Data should be stored in form a table and no other means to stored the data
- III) E.g.If want to stored data of student in the form of table.Consider name of Table is Students , it has four field(i.e column name) Roll_no, Firstname ,Lastname and date_of_birth and Consist five record mans Five rows

Students Table			
Roll_no	Firstname	Lastname	date_of_birth
101	Sachin	Godbole	17/07/1981
102	Mahavir	Jain	04/12/1985
103	Dinesh	Maheshwari	09/10/1987
104	Yogesh	Lad	06/11/1985
105	Mahesh	Thorat	07/06/1989

2) Guaranteed access rule

- I) The guaranteed access All data must be accessible. The Rule say that there is fundamental requirement of primay key for each record in table ,and there should be no ambiguity by stating the table name and its primary key of the each record in the table along with columns name to be accessed rule
- II) For accessing the data from the table , we must provides Table name , Primary key(ie Each unique value for each record(row) in the table) and other column names in the table to be accessed
- III) Considered the above Students table, if we want to find the First name , Lastname and date_of_birth of student whose Roll_no is 103
- IV) Here , the Roll_no is the primary key for the Students Table, This Roll_no Columns is distinct from all other columns,based upon the primary key, all the information present in the table must be guaranteed accessed

3) Systematic Treatment of Null values

- I) Null values could not be treated as blank space or zero values, The null values are known as unknown values, unassigned values should be treated as missing information

and inapplicable information that should be treated as systematic, distinct from regular values

- II) Null values is very important concept in the database, A null value must be represented as missing information in the table, it is not same as the blank space, dash, or zero, hash or any other representation
- III) A null value means that we don't know what information must be provided or entered in to this field name
- IV) Null values must be handled logically and consistent manner

4) Active online catalog based on the relational model:

- I) The system must support an online catalog based data dictionary which hold the information or description about the table in the database
- II) User Tables: The user table contains the data about the table which is created by any users in the database systems
- III) System tables: The system table contains the data about the structure of the database and database object
- IV) Metadata: The data which hold the description of table in the database, the table structure, database structure, the relationship among the tables, the queries and on, This data is often called as metadata, in short term, Metadata is data about the data
- V) The collection of the system tables is known as the system catalogs or data dictionary

5) The comprehensive data sublanguage rule:

I) The system must support at least one relational language that through which the data in the database must be accessed

II) The language must support all the operation of the following items:

- Data definition
- View definition
- Data manipulation
- Integrity constraints
- Authorization
- Transaction boundaries (begin, commit and rollback)

6) The view updating rule:

- I) All the view must be theoretically updatable can be updated by the system
- II) There is ambiguity in this rule, the Structured query language support a single updation at a time suppose if we try combine two or more tables a for a complex views and try to update the views and the DBMS would fail to update the records to the respective tables, thereby violating this rule.
- IV) If that view doesn't include the primary key columns in the view, then each record in the table cannot be updated, thereby violating this rule.

Eg. If Roll_no column is not present in the view then it is not possible update the view of the student table

7) High-level insert, update, and delete:

- I) This rules states that in the relational model, the structured query language must performed data manipulation such as inserting ,updating and deleting record on sets of rows in the table
- II) You expected from the RDBMS, that you can retrieves all the record from table applying single command on the set of tables,or by using single query statement, this rules state that not only retrieves all the record from table but also you can apply the delete , insert, and update multiple records should possible by using the single command
- III) Considered an example, if you want to delete the record of the invoices table which are older than six years,you don't have locate postion each record and delete them individually , uou should able to delete set of records in the table using one single command
- IV) The same concept can be apply to inserting and updating the record

8) Physical data independence:

- i) Any change made in the data is physically stored in term of data is stored in the file system through array and link list must not effect application that access the data structure

This rule says that any change made in the back end (SQL Server/Oracle) must not affect the front end application (Visual Basic/Java).

If the database file is renamed or the database location changes, then this should not have an effect on the application.

9) Logical data independence:

This rule states that changes in the logical level (rows, columns, and so on) must not change the application's structure.

This rule states that it should be possible to change the database design or alter the database design without the user being aware of it.

These changes could be adding a new table to the database or deleting a table from the database, but the application must be unaffected for accessing the data structure.

Consider an example: if you want to improve the performance of searching records in a table, for that reason you might split the Customer table into parts, i.e., Customer_India and Customer_Rest. This allows you to search a record in the Customer_India table rapidly, but what about the existing user who is referring to the Customer table. In practice, it can be done by creating a view which combines two tables into a single table with the same name, so that there should be no effect on the application.

10) Integrity independence:

Data integrity constraints should be considered as separated from the application program, the structured query language which defines data integrity constraints must be stored in the database in terms of data in a table that is, in the catalog and not in the application.

Referential integrity and entity integrity are integral parts of the relational database. In more specific terms, the following two types of integrity should be applied to the relational database.

i) Entity integrity: The column which has the primary key value should not contain missing values or duplicate values. This means the column should not contain null values and must have a unique value in each record set.

ii) Referential integrity: The column which have the foreign key value, there must exist a matching primary key column value mean the foreign key column have duplicate value must be referential to primary key column value.

11) Distribution independence

The rules states that the data can be stored centrally on the single machine or it can be stored in the various location(ditributed) on various machine but it should be invisible to the user i.e the user does not location of data is stored whether on the single machine(Centrally stored) or the distributed stored.If the location of database in change then the existing application must continually access the change database

One of the important benefits of networking is that it allows multi-user access to a database; that is, the users can access the data which is distributed across the network.

However, it is also possible to distribute the data across the same network.

This rule also state that even if the table moves from one location to another location the user should aware of it, it should be transparent to the user, changing in the location mean that the application should not be rewriiten.

12 Nonsubversion rule

The system must not have features that allow you to subvert database structure integrity. Basically, the system must not include back doors that let you cheat the system for features such as administrative privileges or data constraints.

To understand another way, a user should not be allowed access the database by means of other way, other than SQL



RELATIONAL DATABASE MODEL

Unit Structure

6.0 Objectives

6.1 RELATIONAL DATABASE MODEL:

E.F.codd first proposed the relational database Model also he is known as the father of Relational model.

Relation model was attempt to specify the database structure in term of matrix.ie the database should contain tables.The tables is in form of set of Columns and Rows. The relational model is set of 2 dimensional table consists of rows and columns

Tables in the database is known as relation and Columns in the table is called as attributes of an tables and rows in the table is called records or tuples

In the relational database model consists of set of tables having the unique name

One row in the table represents a relationship among the other tables in the database. The set of values in one table is related to the set of values in another table. Thus the table represents a collection of relationships. The relationship among the tables in the form of primary key – foreign key relationship.

6.2 LOGICAL VIEW OF DATA:

1 introduction

Logical structure of tables consists of 2-dimensional tables consisting of numbers of horizontal rows and vertical columns.

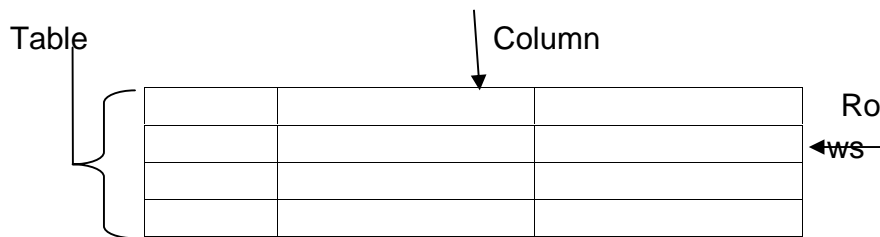


Table is an abstract entity which does not say how the data is stored in the physical memory of the computer system.

Each table in the database has its own unique name through which we can refer to the content of the table by the unique name.

2.Characteristics of an table

I) A table in the database must be in the two-dimensional structure which consists of a number of rows and columns.

II) Each row in a table is called a record or tuple and can represent a single entity which occurs within the entity set. For example, a Customer record in the Customer table.

III) Each column name in the table is called an attribute, and each row in the table is called a record. Each column name in the table is a unique name. For example, no duplicate names in the same table can be repeated.

IV) Each row/column intersection represents a single data item.

V) All the values in the column must be represented in the same data format.

VI) Each column has a specific range of values, and also referred to as the domain attribute.

VII) The order of rows and columns is not limited to the DBMS.

3.Example

There is Customer Table contain all information about the
 Customer
 Cust_id
 Cust_Name
 Cust_Age
 Cust_Address
 Cust_Mobile_No
 Cust_Phone_No

Cust_id	Cust_Name	Cust_Age	Cust_Address	Cust_Mobile_No	Cust_Phone_No
1	Yogesh	20	Worli	9892456123	0224672345
2	Ramesh	23	Bandra	9320896742	0225678894
3	Ram	18	mahim	9819674534	0224678678
4	Pramod	24	Khar Road	9821673445	0223456478
5	Yatin	25	Dadar	9892396735	0222456783
6	Tushar	26	Matunga	9867458432	0226783452

Attribute

- Each column in the above table represent the data item in the database
- Each column in the table represent the attribute in the table
- Atleast one column consist in the table
- There must be one unique column in the table , this means that no two columns has the same name in the same table ,it is possible to have two column with same colmn name but it in the different table.
- The ANSI/SQL Standard does not specify a maximum numbers of rows and columns in the table.

Eg. Cust_id ,Cust_Name ,Cust_Age ,Cust_Address ,Cust_Mobile_No, Cust_Phone_No are the attributes of the Customer Table

Records/Tuples

- A single Record consist all the information of the single entity.
- Each horizontal row in the Customer table represented a single entity
- A Table consist any number of rows, The ANSI/SQL Standard doesnt specify the limits of rows in the table.
- Empty table is called when there is zero row consist in the table

6.3 KEY

Definition

A Column value in the table that uniquely identifies a single record in the table is called key of an table

A attribute or the set of attribute in the table that uniquely identifies each record in the entity set is called a key for that entity set

Types of keys

Simple Key: A key which has the single attribute is known as a simple key

Composite key: A key which consist two or more attributes is called a Composite Key.

Example:

Cust_id is a key attribute of Customer Table it is possible to have a single key for one customer i.e is Cust_id ie Cust_id =1 is only for the Cust_name ="Yogesh" please refer to the Customer Table which is mentioned above.

Types of key	Definition of Key
Super Key	A key is called super key which is sufficient to identify the unique record in the table
Candidate Key	A minimal super key is called Candidate key .A super key has no proper subset of candidate key
Primary Key	A candidate key is chosen as a principal to identify a unique
Secondary Key	
Foreign Key	an Column (or combination of Columns) in the one tables whose values is match the primary key in the another table

Types of key

1 Super Key

A key is called super key which is sufficient to identify the unique record in the table

Customer Table

Cust_id	Cust_Name	Cust_Age	Cust_Address	Cust_Mobile_No	Cust_Phone_No
1	Yogesh	20	Worli	9892456123	0224672345
2	Ramesh	23	Bandra	9320896742	0225678894
3	Ram	18	mahim	9819674534	0224678678
4	Pramod	24	Khar Road	9821673445	0223456478
5	Yatin	25	Dadar	9892396735	0222456783
6	Tushar	26	Matunga	9867458432	0226783452

Example

Here Cust_id attribute of the entity set Customer is uniquely identify Customer entity from another so The Cust_id is the Super key. Another way is, the combination of Cust_id attribute and Cust_Name attribute is the Super key for the Customer Entity set. Only the Cust_Name is not called the Super Key because several customer may have the Same Name

2. candidate key

Defination:

A minimal super key is called Candidate key .A super key has no proper subset of candidate key

Here Minimum attribute of the super key is omitted unwanted attributed of an table that key is sufficient for identifying the unique record in the entity set so it is called as Candidate key

The Candidate key is also known as the primary key

Example

Cust_id	Cust_Name	Cust_Age	Cust_Address	Cust_Mobile_No	Cust_Phone_No
1	Yogesh	20	Worli	9892456123	0224672345
2	Ramesh	23	Bandra	9320896742	0225678894
3	Ram	18	mahim	9819674534	0224678678
4	Pramod	24	Khar Road	9821673445	0223456478
5	Yatin	25	Dadar	9892396735	0222456783
6	Tushar	26	Matunga	9867458432	0226783452

From above statement say combination of Cust_id attribute and Cust_Name is a super key for the Customer entity set it is required to distinguish one record on the Customer entity from another record of same set.

But Cust_id attribute of the Customer entity is also known as minimal super key which is also enough to distinguish one record from customer entity from another record from customer entity set, because Cust_Name is the additional attribute of the Customer table.

2 Primary key**Definition**

Primary key of the table is a column or combination of the some columns whose values uniquely identify a single record in the table.

Primary key state no two records of the table contain the same value in that column or combination of the column. It states that a unique identifier for the entity set.

Cust_id	Cust_Name	Cust_Age	Cust_Address	Cust_Mobile_No	Cust_Phone_No
1	Yogesh	20	Worli	9892456123	0224672345
2	Ramesh	23	Bandra	9320896742	0225678894
3	Ram	18	mahim	9819674534	0224678678
4	Ram	20	Khar Road	9821673445	0223456478
5	Yatin	25	Dadar	9892396735	0222456783
6	Tushar	26	Matunga	9867458432	0226783452

In above table the Customer Age cannot act as primary key hence the customer age column contain repeated values and Customer Name also cannot act as primary key because it earlier state that several customer may have the same name hence Cust_Name column has the repeated values .

Hence there Cust_id can act as the primary key in the Customer table this is only column which contain a unique set of values.

3 Secondary key

Defination

Secondary key of the table consist the column and combination of the some columns which meant for data retrieval purpose.

The secondary key not always required to primary key, other tah the pimary key there are some attribute which is required to retrieve data from the customer table using the another attribute such as Cust_Name and Cust_Age columns

Cust_id	Cust Name	Cust Age	Cust_ Address	Cust_Mobile_ No	Cust_Phone_ No
1	Yogesh	20	Worli	9892456123	0224672345
2	Ramesh	23	Bandra	9320896742	0225678894
3	Ram	18	mahim	9819674534	0224678678
4	Ram	20	Khar Road	9821673445	0223456478
5	Yatin	25	Dadar	9892396735	0222456783
6	Tushar	26	Matunga	9867458432	0226783452

In the above Customer Table Cust_Name and Cust_Age attribute act as the Secondary key

Foreign Key

A Column (or combination of Columns) in the one tables whose values is match the primary key in the another table is called as a foreign key

Foreign key can also have one or more column like as primary key

A single table may contain more than one foreign key which is related to the more tah one table, the table which used the foreign key is said the referential integrity

What the referential integrity

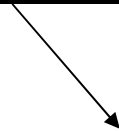
Referential integrity say the column which contain foreign key in one table must be primary key of another table

In general term, Foreign key of Table A must be Primary key of Table B

Example

Customer Table

Cust_id	Cust_Name	Cust_Age	Cust_Address	Cust_Mobile_No	Cust_Phone_No
---------	-----------	----------	--------------	----------------	---------------



Account Table

Account No	Cust_id	Account type	Balance	Description
------------	---------	--------------	---------	-------------

In The above example Cust_id is the primary key for the customer Table while Cust_id is the foreing key for the Account table

Here the the datatype assigned to column and Number of column in the foreign key is same as to the primary key.

Cust_id	Cust_Name	Cust_Age	Cust_Address	Cust_Mobile_No	Cust_Phone_No
1	Yogesh	20	Worli	9892456123	0224672345
2	Ramesh	23	Bandra	9320896742	0225678894
3	Ram	18	mahim	9819674534	0224678678
4	Ram	20	Khar Road	9821673445	0223456478
5	Yatin	25	Dadar	9892396735	0222456783
6	Tushar	26	Matunga	9867458432	0226783452

Account No	Cust_id	Account type	Balance	Description
101	1	Saving	10,000	
102	2	saving	20,200	
103	2	Saving	20,200	
104	3	Current	11,000	
105	4	Saving	50,000	

6.4 RELATIONAL INTEGRITY RULES

1) Entity Integrity

Entity Integrity ensure that there is no duplicate records in the table and each field that recognizes each record in the table must have unique value and not having null values

Entity Integrity specifies that every instance of entity have the unique values ie primary key must be kept and must have the values other than null values.

Entity Integrity is the mechanism the Database management system provides to maintain primary keys. The primary key is known as unique identifier for each rows in the table . Entity Integrity must have two properties for primary keys:

- The primary key must be unique for each row in the table that is no two primary key having the same value in the same table, The primary key values must be distinct i.e the value could not be repeated.
- The primary key values should not contain null values, primary key must be NOT NULL

The uniqueness property ensures that the primary key of each row uniquely identifies it; there are no duplicates. The second property ensures that the primary key has meaning, has a value; no component of the key is missing.

2. Referential Integrity

Referential integrity is a property of data which, when satisfied, requires every value of one attribute (column) of a relation (table) to exist as a value of another attribute in a different (or the same) relation (table).

For referential integrity to hold in a relational database, any field in a table that is declared a foreign key can contain only values from a parent table's primary key or a candidate key. For instance, deleting a record that contains a value referred to by a foreign key in another table would break referential integrity.

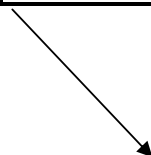
Some relational database management systems (RDBMS) can enforce referential integrity, normally either by deleting the foreign key rows as well to maintain integrity, or by returning an error and not performing the delete.

Foreign key

A column or collection of column in one table whose values must match the primary key in the other table is known as a foreign key

Cust_id	Cust_Name	Cust_Age	Cust_Address	Cust_Mobile_No	Cust_Phone_No
1	Yogesh	20	Worli	9892456123	0224672345
2	Ramesh	23	Bandra	9320896742	0225678894
3	Ram	18	mahim	9819674534	0224678678
4	Ram	20	Khar Road	9821673445	0223456478
5	Yatin	25	Dadar	9892396735	0222456783
6	Tushar	26	Matunga	9867458432	0226783452

Account No	Cust_id	Account type	Balance	Description
101	1	Saving	10,000	
102	2	saving	20,200	
103	2	Saving	20,200	
104	3	Current	11,000	
105	4	Saving	50,000	



In above example

Cust_id column of Account Table is foreign key for the Account table while it is primary key for the Customer Table

3. Other integrity rules

NOT NULL. As the integrity rules states column which specify the NOT NULL values mean these column must contain some values which should not contain any NULL values

Unique. In this rules no two record or tuples have same values for the same attribute

Check. In this rule we can apply own integrity rules by applying CHECK Constraint.

6.5 RELATIONAL DATABASE DESIGN PROCESS

The Relational Database model was proposed by E.F.Codd in 1969. The Relational Database Model is based on branch of mathematics called set theory and predicate logic. The idea behind to design the Relational Database model is that the database consist of series of unordered table or relation that can be manipulated using non-procedural process that return tables

Note: it is Commonly thought that word relational in the relational model comes from the fact that the tables is related together in the relational model, but it is inconvenient way to think of the term , but it is not accurate.. The table which codd is specifies while in writings was actually referred to as relation (a related set of Information).

While designing relational database model you have consider in the mind that how choose a best model in the real world and how this best model is fitted in the database. While designing the relational model you have o consider that which table you want to create, what column the table will consist, consider the relationship between the tables. While developing the relational model it would be nice you process was totally clear and intuitive or it can be even better to automated.

- The benefits of a relational Database Design process.
- Data entry, updating and deleting would be efficient and simple in manner
- Data retrieval, summarization and reporting will be efficient

- Database must follow a well designed model hence it behaves predictably
- Large amount of information must be stored in the database rather than in the application, the database must be somewhat well documented
- Changes to database structures are easy to make e.r creating database, tables, views.

6.5.1 Feature of Good Relational Database Design-Normalization

- In the Relational Database Design, the process of organizing data to minimize redundancy is known as Normalization
- The main aim of the Normalization is to decompose complex relations into smaller, well-structured relations
- Normalization is the process that involves dividing a large table into smaller tables (which contain less redundant data) and stating the relationship among the tables.
- Data normalization or Database Normalization is also canonical synthesis, it means preventing inconsistency in a set of data by using unique values to reference common information
- The main objective of the normalization is to isolate the data so that users can apply operations such as addition, deletion and modification of a field in one table and then its propagated to the rest of the database through the well defined relationships
- The same set of data is repeated in multiple tables of database so there are chances that data in the database may lead to be inconsistent, so while updating, deleting or inserting the data into the inconsistent database which leads to a problem of data integrity
- If we can apply the normalization on the table we can reduce the problem of data inconsistency to some extent

Definition of Normalization

In the Relational Database Design, the process of organizing data to minimize redundancy is known as Normalization

Main aim of the Normalization

- 1. Ensure data integrity**
 - The correct data should be stored in the database

- ii) This can be achieved by applying integrity rules in the database
- iii) Integrity rules prevent duplicate values in the database

2. Prevent Data Redundancy in database

- i) Non-Normalized data is more vulnerable to data anomalies. The same set of information is present in the multiple rows, now if we applying the updating rule on the table then it lead to logical inconsistency this is known as update anomaly

An insufficiently normalized table might have one or more of the following characteristics:

update anomaly

The same set of information is present in the multiple rows, now if we applying the updating rule on the table then it lead to logical inconsistency. Consider an example of customer Table which contain set of attributes such as Cust_id ,Cust_Name, Cust_Address,

Cust_id	Cust_Name	Cust_Address
423	Pramod	Nerul
423	Pramod	Nerul
567	Manish	Vashi
567	Manish	Bhandup

Thus a change of address of a particular Customer will need update to multiple records. If the update is not carried out successfully—if, that is, the Customer's address is updated on some records but not others—then the table is remains in an inconsistent state. Specifically, the table provides conflicting answers to the question of what this particular customer's address is. This Known is known as an **update anomaly**.

The above Customer Table is Cust_id =567 having different address in the multiple records

An insertion anomaly

There are some circumstances in which certain fact cannot recorded at all

Example Consider a table Faculty and Course_code consist the Column name

Faculty_ID, Faculty_Name, Faculty_Hire_Date, Course_Code

Faculty_ID	Faculty_Name	Faculty_Hire_Date	Course_Code
386	Mahesh Lad	10/06/1994	ENG-207
197	Jayesh Shinde	12/06/1987	PP-205
197	Jayesh Shinde	12/06/1987	PP-206
234	Pramod Bhave	11/07/2005	?

Thus we can add the record the details of any faculty member who teaches at least one course, but we cannot record the details of a newly-hired faculty member who has not yet been assigned to teach any courses except by setting the Course Code to null. This known as an **insertion anomaly**.

In the above Table Until the new faculty member, Pramod Bhave , is assigned to teach at least one course, his details cannot be recorded.

An deletion anomaly.

There are circumstances in which the deletion of data representing certain facts necessitates the deletion of some unrelated data . The "Faculty and Courses" table suffers from this type of anomaly, for if a faculty member temporarily ceases to be assigned to any courses, we must delete the last of the records on which that faculty member appears, effectively also deleting the faculty member. This is known as a **deletion anomaly**.

Faculty_ID	Faculty_Name	Faculty_Hire_Date	Course_Code
386	Mahesh Lad	10/06/1994	ENG-207
197	Jayesh Shinde	12/06/1987	PP-205
197	Jayesh Shinde	12/06/1987	PP-206

Delete

All information about Mahesh Lad is lost when he temporarily ceases to be assigned to any courses.

Advantage of Normalization

- 1) Avoids data modification (INSERT/DELETE/UPDATE) anomalies as each data item lives in One place
- 2) Greater flexibility in getting the expected data in atomic granular
- 3) Normalization is conceptually cleaner and easier to maintain and change as your needs change
- 4) Fewer null values and less opportunity for inconsistency
- 5) A better handle on database security
- 6) Increased storage efficiency
- 7) The normalization process helps maximize the use of clustered indexes, which is the most powerful and useful type of index available. As more data is separated into multiple tables because of normalization, the more clustered indexes become available to help speed up data access.

Disadvantage of Normalization

- 1) Requires much more CPU, memory, and I/O to process thus normalized data gives reduced database performance
- 2) Requires more joins to get the desired result. A poorly-written query can bring the database down
- 3) Maintenance overhead. The higher the level of normalization, the greater the number of tables in the database

6.6 NORMAL FORM

Normal form are designed for addressing potential problem in the database such that inconsistent and redundant data which is stored in the database

Normal form is based on relation rather than table . The normal form has a set of attribute which table should be satisfy. The Following attributes are

- 1) They describe one entity
- 2) They do not have duplicate rows, hence there must a primary key for each row.
- 3) The columns are unordered
- 4) The rows are unordered

Types of Normal Forms

- 1) Edgar F. Codd, the inventor of the relational model, introduced the concept of normalization and what we now know as the First Normal Form (1NF) in 1970.
- 2) Second Normal Form (2NF) and Third Normal Form (3NF) in 1971,
- 3) Codd and Raymond F. Boyce defined the Boyce-Codd Normal Form (BCNF) in 1974.
- 4) Fourth normal form(4NF)
- 5) Fifth Normal form(5NF)
- 6) Higher normal forms were defined by other theorists in subsequent years, the most recent being the Sixth Normal Form (6NF) introduced by Chris Date, Hugh Darwen, and Nikos Lorentzos in 2002.

6.6.1 First Normal Form

This Normal form is introduced by Edgar F. Codd, is Known as First Normal Form(1NF) in 1971

Definition

A relational database table which consist first normal form (1NF) is to meets certain minimum set of criteria. These criteria are basically concerned with ensuring that the table is a faithful representation of a **relation** and that it is free of **repeating groups**

1. There are no duplicated rows in the table.
2. Each cell is single-valued (i.e., there are no repeating groups or arrays).
3. Entries in a column (attribute, field) are of the same kind.

Let us consider the example

Consider a table "Customer_Rental" consisting the attribute such as Customer_NO, Cust_Name Property_no, P_Address, Rent_start, Rent_finish, Rent, Owner_No, Owner_Name

Customer_NO	Cust_Name	Property_no	P_Address	Rent_start	Rent_finish	Rent	Owner_No	Owner_Name	
CR78	Mahesh Lad	PG34	Nerul, Navi Mumbai	1-July-91	30-Oct-95	450	C045	Sanjay More	
		PG78		Turbhe, Navi Mumbai	1-Nov-95	1-Nov-98	500	C093	Mahavir Jain
CR98	Pramod Patel	PG34	Nerul, Navi Mumbai	1-July-95	30-Oct-98	450	C045	Sanjay More	
		PG36		Kalyan, Thane	1-Nov-97	1-Nov-99	350	C093	Mahavir Jain
		PG78		Karjat, Raigad	1-july-96	1-Nov-97	450	C093	Mahavir Jain

The above table does not contain the atomic values in the Property_no, P_Address, Rent_start, Rent_finish, Rent, Owner_No, Owner_Name. Hence it is called un-normalized table, we cannot insert, update and delete the record from the table because it is in an inconsistent state. The above table has to be normalized.

Customer_NO	Cust_Name	Property_no	P_Address	Rent_start	Rent_finish	Rent	Owner_No	Owner_Name
CR78	Mahesh Lad	PG34	Nerul, Navi Mumbai	1-July-91	30-Oct-95	450	C045	Sanjay More
CR78	Mahesh Lad	PG78	Nerul, Navi Mumbai	1-Nov-95	1-Nov-98	500	C093	Mahavir Jain
CR98	Pramod Patel	PG34	Nerul, Navi Mumbai	1-July-95	30-Oct-98	450	C045	Sanjay More
CR98	Pramod Patel	PG36	Kalyan, Thane	1-Nov-97	1-Nov-99	350	C093	Mahavir Jain
CR98	Pramod Patel	PG78	Karjat, Raigad	1-july-96	1-Nov-97	450	C093	Mahavir Jain

The above table show the same set of data as the previous table however we have eliminated the repeated groups.so the table shown in the above table to be in First Normal form(1NF)

6.6.2 Second Normal Form

Second Normal Form based on the concept of Full Functional Dependency and it tries remove the problem of redundant data in the First normal form.

Defination: A 2NF relation in 1NF and every non-primary key attritube is fully functionally dependent on the primary key

Converting from 1NF to 2NF:

- o Firstly Identify the primary key for the 1NF relation.
- o Identify whether the functional dependencies in the relation.
- o If partial dependencies exist on the primary key remove them by placing then in a new relation along with a copy of their determinant.

Example

Functional Dependency for Customer_Rental Relation

Step 1 : Primary key: Customer_No + Property_no

Step 2 :Full Functional Dependency:

(Customer_No+Property_No)->(Rent-Start, RentFinish)

Step3 Partial Dependency:

(Customer_No+Property_No)->Cust_Name

(Customer_No+Property_No)->(P_Address, Rent, Owner_No, Owner_Name)

<u>Customer</u> _NO	Cust_ Name	<u>Property</u> no	P_Address	Rent_start	Rent_finish	Rent	Owner_ No	Owner_ Name
------------------------	---------------	-----------------------	-----------	------------	-------------	------	--------------	----------------

Customer Relation

Customer_NO	Cust_Name
CR78	Mahesh Lad
CR98	Pramod Patel

Rental Relation

Customer_NO	Property_No	Rent_start	Rent_finish
CR78	PG34	1-July-91	30-Oct-95
CR78	PG78	1-Nov-95	1-Nov-98
CR98	PG34	1-July-95	30-Oct-98
CR98	PG36	1-Nov-97	1-Nov-99
CR98	PG78	1-july-96	1-Nov-97

Property_owner Relation

Property_No	P_Address	Rent	Owner_No	Owner_Name
PG34	Nerul, Navi Mumbai	450	C045	Sanjay More
PG78	Nerul, Navi Mumbai	500	C093	Mahavir Jain
PG36	Kalyan, Thane	350	C093	Mahavir Jain

Here Customer_no is the only key to identify The Customer name hence Customer_No is the primary key in the Customer Relation Table but Foreign key in the Rental relation table

6.6.3 Third Normal Form

Third Normal form Based on the concept of transitive dependency.

A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.

Converting from 2NF to 3NF:

- o Identify the primary key in the 2NF relation.
- o Identify functional dependencies in the relation.
- o If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their dominant

Property_Owner to 3NF Relations

Property_owner Relation

<u>Property_No</u>	P_Address	Rent	Owner_No	Owner_Name
--------------------	-----------	------	----------	------------

Transitive Dependency:

(Customer_No+Property_No)->Owner_No

Owner_No ->OName

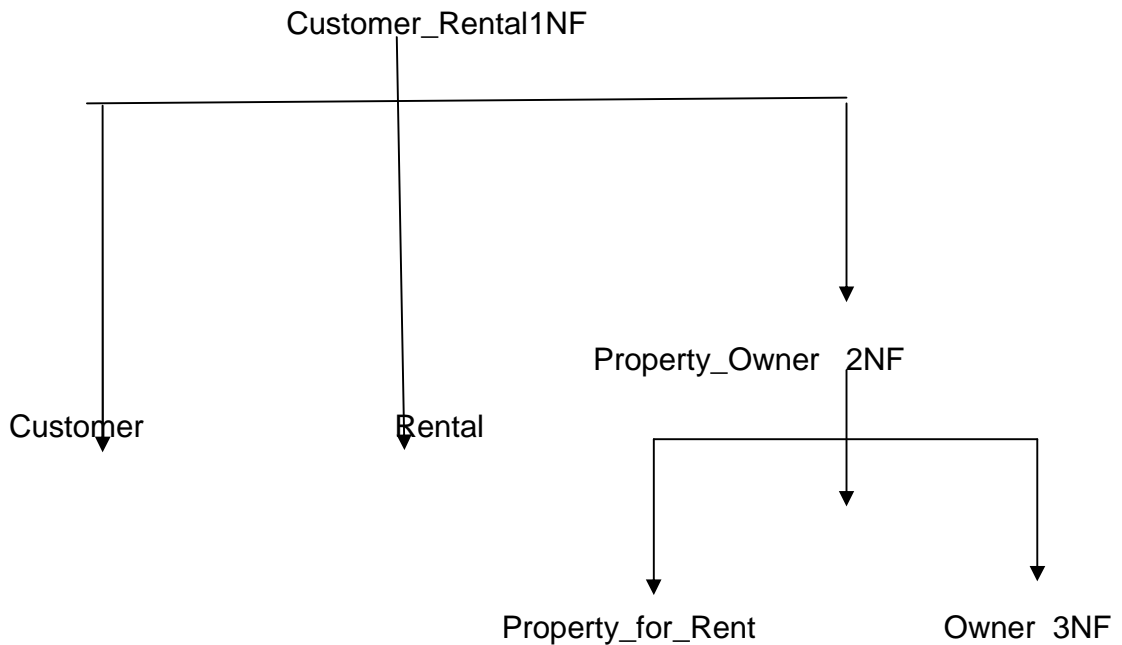
Property_for_Rent

<u>Property_No</u>	P_Address	Rent	Owner_No
PG34	Nerul,Navi Mumbai	450	C045
PG78	Nerul,Navi Mumbai	500	C093
PG36	Kalyan,Thane	350	C093

Owner

<u>Owner_No</u>	Owner_Name
C045	Sanjay More
C093	Mahavir Jain

Process of Decomposition

**6.6.4 Boyce-Codd Normal Form (BCNF)**

o Based on functional dependencies that takes into account all candidate keys in a relation.

o For a relation with only one candidate key, 3NF and BCNF are equivalent.

o A relation is in BCNF, if and only if every determinant is a candidate key.

o Violation of BCNF may occur in a relation that

- contains 2 (or more) composite keys
- which overlap and share at least 1 attribute

3NF to BCNF

o Identify all candidate keys in the relation.

o Identify all functional dependencies in the relation.

o If functional dependencies exists in the relation where their determinants are not candidate keys for the relation, remove the

functional dependencies by placing them in a new relation along with a copy of their determinant.

Example - 3NF to BCNF Relations

Client_Interview Relation

Client_No	Interview_Date	Interview_Time	Staff_No	Room_No
CR76	13/05/98	10.30	SG5	G101
CR56	13/05/98	12.30	SG5	G101
CR74	13/05/98	12.30	SG37	G102
CR56	01/06/08	10.30	SG5	G102

(Client_No, Interview_Date) → (Interview_Time, Staff_No, Room_No)

(Staff_No, Interview_Date, Interview_Time) → Client_No

(Room_No, Interview_date, Interview_Time) → Staff_No, Client_No

(Staff_No, Interview_Date) → Room_No

Client_No	Interview_Date	Interview_Time	Staff_No
CR76	13/05/98	10.30	SG5
CR56	13/05/98	12.30	SG5
CR74	13/05/98	12.30	SG37
CR56	01/06/08	10.30	SG5

Staff_No	Interview_Date	Room_No
SG5	13/05/98	G101
SG37	13/05/98	G102
SG5	01/06/08	G102



INTRODUCTION TO UML

Unit Structure

- 7.0 Objectives
- 7.1 Introduction

1.0 OBJECTIVES

UML or Unified Modeling Language is a specification which is used in the software engineering field. It can be defined as a general purpose language which is used to design as graphical notation which is used as an abstract model and this abstract model is used in the system. That system is called as UML or Unified Model language.

Why Modeling is required and what is the principle of Model?

Analysis the problem domain that is simply reality captures requirements in the design the model, visualize the system in its entirety, and specify the structure and / or behavior of the system

Choose your model well

The choice of model such way that it should be through analysis of the problem and the design of the solution.

Every model in the system can be expressed at different levels of accuracy - the same model can be scaled up (or down) to different granularities.

The best models in the system are closer to reality - simplify the model as much as possible and don't hide important details.

No single model suffices - every nontrivial system has different set of dimensions to the problem and has much solution

UML is an modeling Language but not a methodology or process , the first concept is developed by Grady Booch , James Rumbaugh and Ivar Jacobson at Rational Software.

This model is accepted as a standard by the Object Management Group (OMG), in 1997

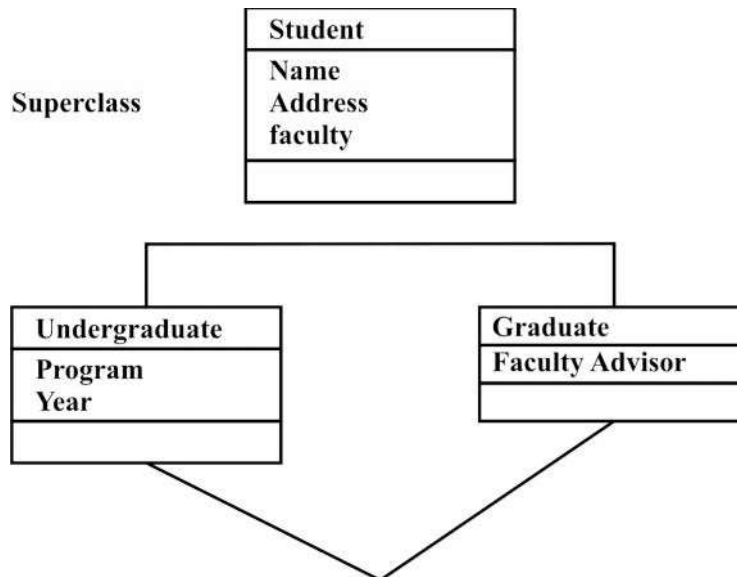
7.1 TYPE OF UML

The Main purpose of the class diagram is include the classes within a model. In the object oriented programming , the classes has certrian attributes(i.e data member) , operations(member function) and relationship among the objects , In the UML the class diagram can be include very easily. The fundamental part of the class diagram is the class icon which can represented a class. The class icon which is shown in the figure

Class
attributes
Member function

A class icon is simply a rectangle divided into three compartments. The topmost compartment contains the name of the class. The middle compartment contains a list of attributes (member variables), and the bottom compartment contains a list of operations (member functions). In many diagrams, the bottom two compartments are omitted. Even when they are present, they typically do not show every attribute and operations. The goal is to show only those attributes and operations that are useful for the particular diagram.

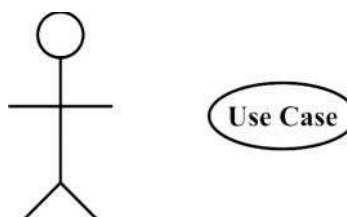
If two classes are very similar it may be helpful to put the similarities into a more general class called a **superclass**. For example, if you set up a superclass called Student, then Graduate Student and Undergraduate Student can be subclasses of Student.



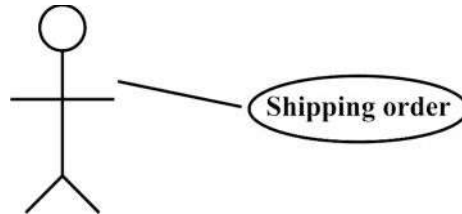
7.2 USECASE DIAGRAM

A use case is a set of scenarios that shows an interaction between a user and a system. A use case diagram shows the relationship among actors and use cases.

The two main components of a use case diagram are use cases and actors.



An actor is represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.



7.3 ACTIVITY DIAGRAMS

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

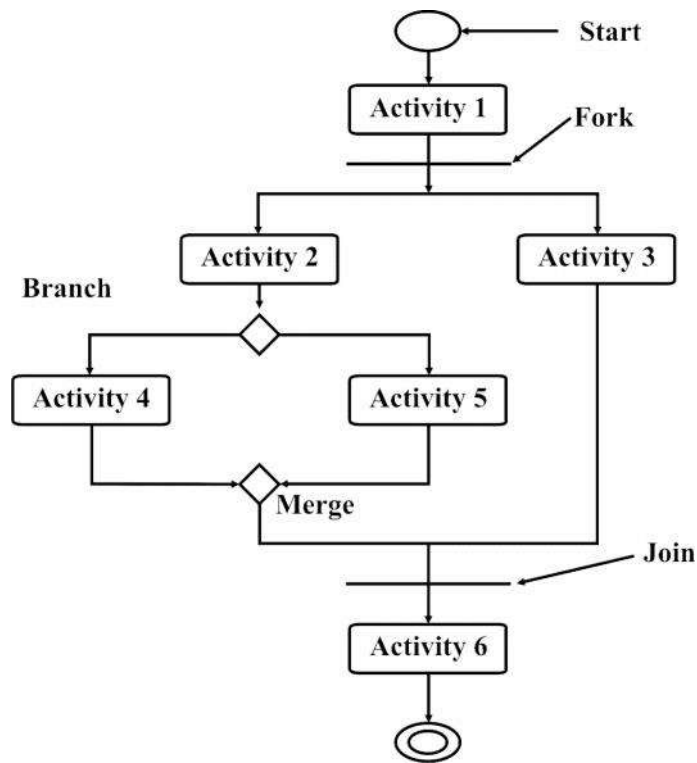
Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- rounded rectangles represent activities;
- diamonds represent decisions;
- bars represent the start (split) or end (join) of concurrent activities;
- a black circle represents the start (initial state) of the workflow;
- an encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen.

Hence they can be regarded as a form of flowchart. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

5



7.4 SEQUENCE DIAGRAMS

Sequence diagrams is involved how to the object are interacted which are arranged in a time sequence. The Sequence Diagram which is use the flow of events to determine what objects and interactions I will need to accomplish the functionality specified by the flow of events.

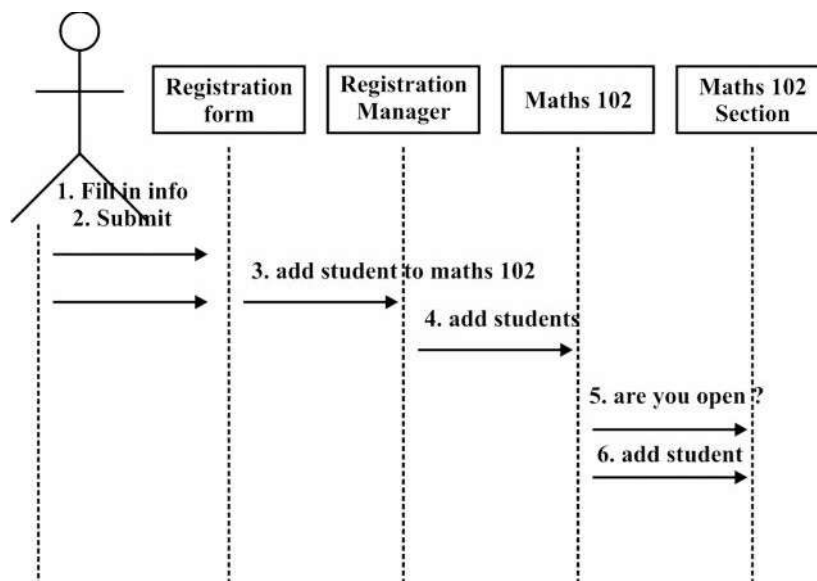
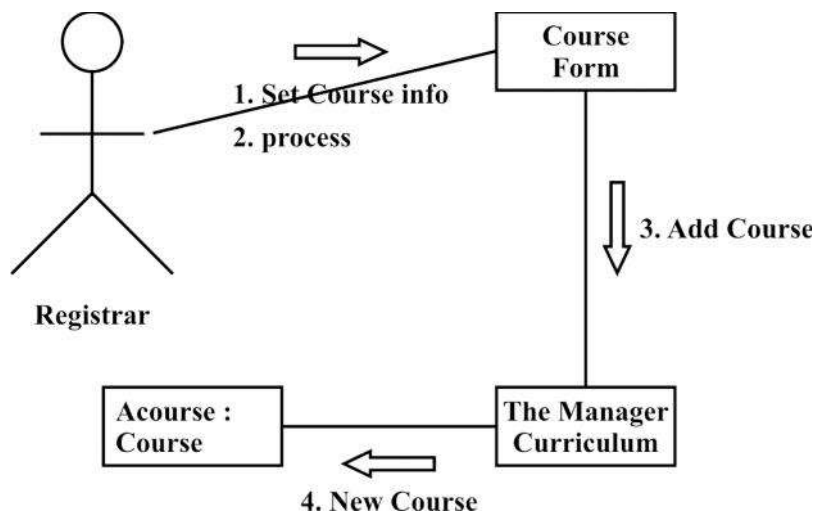


Figure shows how a student successfully gets added to a course. The student let's call him Mahesh) fills in some information and submits the form. The form then talks to the manager and says "add Joe to Mahesh 102 ." The manager tells Math 102 that it has to add a student.

Math 102 says to Section 1 "are you open?" In this case, Section 1 replies that they are open, so Math 103 tells section 1 to add this student. Again, sequence diagrams are great tools in the beginning because they show you and your customer step-by-step what has to happen.

7.5 COLLABORATION DIAGRAMS

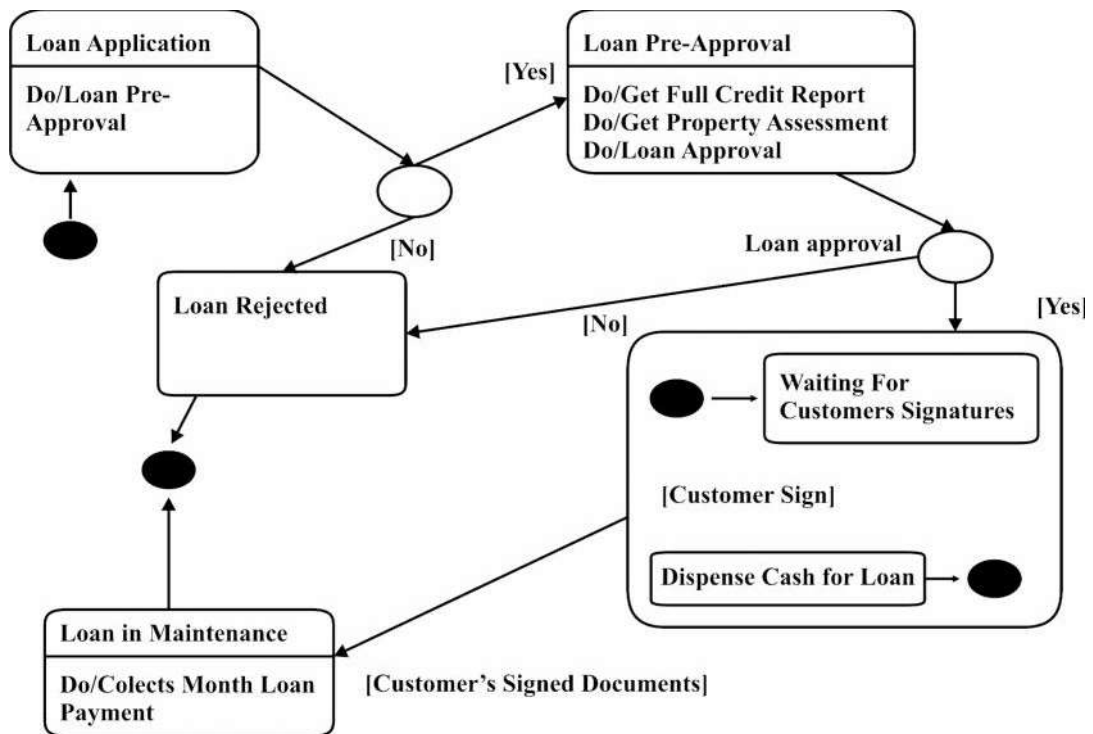


7.6 STATECHART DIAGRAM

The statechart diagram models the different states that a class can be in and how that class transitions from state to state. It can be argued that every class has a state, but that every class shouldn't have a statechart diagram. Only classes with "interesting" states -- that is, classes with three or more potential states during system activity -- should be modeled.

As shown in Figure 5, the notation set of the statechart diagram has five basic elements: the initial starting point, which is drawn using a solid circle; a transition between states, which is drawn using a line with an open arrowhead; a state, which is drawn using a rectangle with rounded corners; a decision point, which is

drawn as an open circle; and one or more termination points, which are drawn using a circle with a solid circle inside it. To draw a statechart diagram, begin with a starting point and a transition line pointing to the initial state of the class. Draw the states themselves anywhere on the diagram, and then simply connect them using the state transition lines.



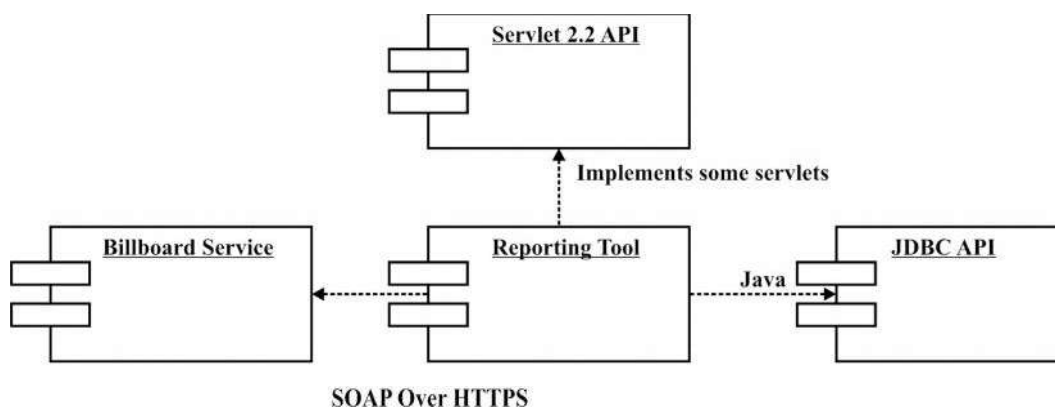
The example statechart diagram in Figure shows some of the vital information they can communicate. For example, you can tell that loan processing department to begin in the Loan Application state. When the pre-approval process is over, depending whatever output comes, and then you move to either the Loan Pre-approved state or the Loan Rejected state. This decision, which is made during the transition process, is shown with a decision point -- the empty circle in the transition line. By looking at the example, a person can tell that a loan cannot go from the Loan Pre-Approved state to the Loan in Maintenance state without going through the Loan Closing state. Also, by looking at our example diagram, a person can tell that all loans will end in either the Loan Rejected state or the Loan in Maintenance state.

7.7 COMPONENT DIAGRAM

A component diagram provides a physical view of the system. Its purpose is to show the dependencies that the software

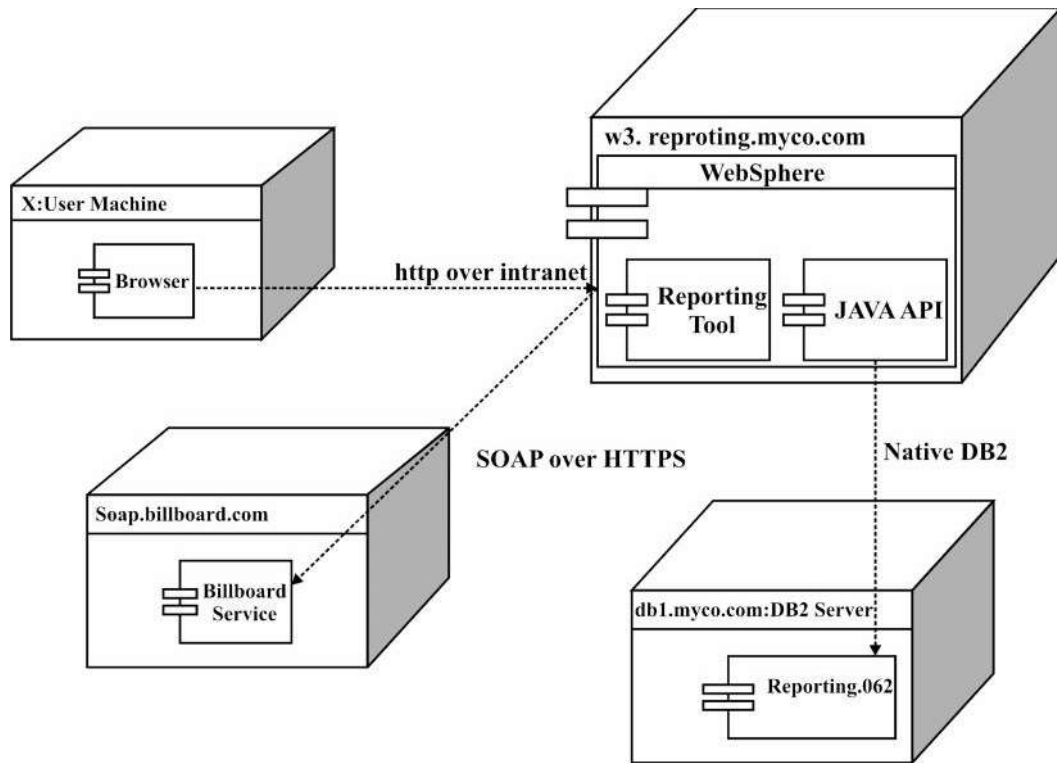
has on the other software components (e.g., software libraries) in the system. The diagram can be shown at a very high level, with just the large-grain components, or it can be shown at the component package level.

Modeling a component diagram is best described through an example. Figure shows four components: Reporting Tool, Billboard Service, Servlet 2.2 API, and JDBC API. The arrowed lines from the Reporting Tool component to the Billboard Service, Servlet 2.2 API, and JDBC API components mean that the Reporting Tool is dependent on those three components.



7.8 DEPLOYMENT DIAGRAM

The deployment diagram shows how a system will be physically deployed in the hardware environment. Its purpose is to show where the different components of the system will physically run and how they will communicate with each other. Since the diagram models the physical runtime, a system's production staff will make considerable use of this diagram. The notation in a deployment diagram includes the notation elements used in a component diagram, with a couple of additions, including the concept of a node. A node represents either a physical machine or a virtual machine node (e.g., a mainframe node). To model a node, simply draw a three-dimensional cube with the name of the node at the top of the cube. Use the naming convention used in sequence diagrams: [instance name] : [instance type] (e.g., "w3reporting.myco.com : Application Server").



The deployment diagram in Figure shows that the users access the Reporting Tool by using a browser running on their local machine and connecting via HTTP over their company's intranet to the Reporting Tool.

This tool physically runs on the Application Server named w3reporting.myco.com. The diagram shows the Reporting Tool component drawn inside of IBM Web Sphere, which in turn is drawn inside of the node w3.reproting.myco.com.

The Reporting Tool connects to its reporting database using the Java language to IBM DB2's JDBC interface, which then communicates to the actual DB2 database running on the server named db1.myco.com using native DB2 communication.

In addition to talking to the reporting database, the Report Tool component communicates via SOAP over HTTPS to the Billboard Service.



RELATIONAL ALGEBRA

Unit Structure

8.0 Objectives

8.0 DATA MANIPULATION LANGUAGES

In order to make the database more useful, then it should be possible to store information in database or retrieve the information from the database. This important role is performed by database Manipulation Language

There are two types of Data Manipulation language

-Navigational (Procedural)

- The query specifies (to some extent) the strategy used to find the desired result e.g. relational algebra


-Non-navigational (non-procedural)

- The query only specifies what data is wanted, not how to find it e.g. relational calculus.

8.1 INTRODUCTION

- Codd proposed a number of algebraic operation for the relational database model
- In the Relation algebra there are two type of operation one is Unary operation and second one Binary operation
- Unary operation takes as input a single table and produces an output another table
- Binary operations take as input two tables and produce as output another table

Fundamental operation

- Unary operation
 - Projection operation()
 - Select Operation()
 - Rename Operation()
- Binary Operation
 - SET operation
 - Union operation(\cup)
 - Difference Operation(-)
 - Intersection Operation()
 - Join Operation()
 - Cartesian Product Operation(X)
 - Division Operation(%)

8.1.1 Selection operation

The Selection operator select the row from the table that satisfy a given predicate. This operation allows to manipulate data in the single relation. The Selection operation is defined by the symbol called sigma(σ). The predicate is appear at subscript of Sigma symbol(σ). The argument relation is present in the parenthesis after the

Syntax

<predicate><Comparsion_operator><Constant_value><(input_table_name)>

Where Predicate: Name of the column in the table

Comparsion_Operator: =, <, <=, >, =>, <>

Example

Select all the student from the student table who's Roll no is greater than 300

Student

Roll No	Students_Name	Students_Address
134	Mary	3 Curry Road
356	John	4 Dockyard
500	Steve	6 Nepean Sea Road

RollNo > 300(Student)

Roll No	Students_Name	Students_Address
356	John	4 Dockyard
500	Steve	6 Nepean Sea Road

We can combine several predicates into a larger predicate by using the connectives and (\wedge), or (\vee), and not (\neg).

To find the tuple in the student table where Student name is john and roll no is greater than 300

Students_Name="John" \wedge RollNo > 300(Student)

Roll No	Students_Name	Students_Address
356	John	4 Dockyard

8.1.2 Projection Operation()

This operator is used to select some of the attributes from the table to produces a desired result set

Note that Projection operation is used to eliminates the duplicates records in the table

Syntax

<attributes>(<Input_Table_Name>)

Example

- 1) Find the all record from the Student table

Roll No, Students_Name, Students_Address (**Student**)

Roll No	Students_Name	Students_Address
134	Mary	3 Curry Road
356	John	4 Dockyard
500	Steve	6 Nepean Sea Road

- 2) Find the Roll no, student name and student address whose rollno is greather than 300

Roll No, Students_Name, Students_Address

(RollNo>300(**Student**))

8.1.3 Rename Operator()

Rename operation gave alternate name to the given column or to any table by using the operator called Rename operator

This operator is used for selecting some specific column from multiple table(set of two or more tables) containing multiple columns having same column name

Rename operator is denoted by the greek letter rho()

Syntax

<New Name for Column>(<Input_Table_Name>)

- 1) Find the all record from the Student table

e.Roll No, e.Students_Name, e.Students_Address
(e(**Student**))

e.Roll No	e.Students_Name	e.Students_Address
134	Mary	3 Curry Road
356	John	4 Dockyard
500	Steve	6 Nepean Sea Road

- 2) Find

8.2 BINARY OPERATIONS

- Two relations are (union) compatible if they have the same set of attributes.
- Example, one table may represent suppliers in one country, while another table with same schema represents suppliers in another country.
- For the union, intersection and set-difference operations, the relations must be compatible.

Union, Intersection, Set-difference

- $R1 \cup R2$
 - The union is the table comprised of all tuples in R1 or R2.
- $R1 \cap R2$
 - The intersection is the table comprised of all tuples in R1 and R2
- $R1 - R2$
 - The set-difference between R1 and R2 is the table consisting of all tuples in R1 but not in R2.

8.2.1 Union Operator

Union operator is used combine all the result form the first query to the result from the second query

Union operator doesnt eliminate duplicate record from the database and they prints the result expression

Syntax

$(\text{Relation1}) \cup (\text{Relation 2})$

Example

1) Employee table

EMPNO	DEPTNAME	EMPFIRSTNAME	EMPLASTNAME
101	Sales	Jayesh	Shinde
102	R&D	Preetesh	Shinde
103	Marketing	Ganesh	lad
104	Sales	Pooja	Lad

2) Project Table

PROJECTNO	DEPTNAME	EMPNO
P1	R&D	103
P2	Sales	104
P3	HR	105

3 Union of the Two Table result in Employee Table and Project Table

Syntax: select deptname from Employee union select deptname from Project;

Result: The return value would be sales, marketing, R&D and HR.

8.2.2 Intersect Operator

This operator is find out all the tuples that all the Common tn the result of Relation 1 and in the Result of Relation 2
Intersect operator doesnot eliminate duplicate record from the database and they prints the result expression

Syntax

- $R1 \cap R2$

Example : Get all the employee's full name that are working on a project.

Syntax: select EMPFIRSTNAME, EMPLASTNAME from Employee, Project where Project. EMPNO =Employee. EMPNO;

Result : Ganesh lad Pooja Lad

8.2.3 Difference Operation

The difference builds a relation consisting of all tuples appearing in the first and not the second of two specified relations. The difference between two relation R1 and R2, R1 MINUS R2, is the set of all tuples belonging to R1 and not to R2.

Syntax R1-R2

Example : Find the employee that are in sales department and are not on project P2.

Syntax: select EMPNO from Employee where DEPTNAME ='Sales' minus
select EMPNO from Project where PROJECTNO ='P2';

Result:EMPNO=101

8.3 CARTESIAN PRODUCT

- $R1 \times R2$
 - The Cartesian product is the table consisting of all tuples formed by concatenating each tuple in R1 with a tuple in R2, for all tuples in R2.
- The Cartesian Product is also an operator which works on two sets. It is sometimes called the CROSS PRODUCT or CROSS JOIN.
- It combines the tuples of one relation with all the tuples of the other relation.

Example of a Cartesian Product

R1

<u>A</u>	<u>B</u>
1	x
2	y

R2

<u>C</u>	<u>D</u>
a	s
b	t
c	u

R1XR2

A	B	C	D
1	x	a	s
1	x	b	t
1	x	c	u
2	y	a	s
2	y	b	t
2	y	c	u

**Example
Employee Table**

Empno	Empname	Deptno
101	Ramesh	100
102	Suresh	200
103	Rajesh	100

Department table

Deptno	Deptname
100	Sales
200	R &D

When we Join the Two table cross product

e.Empno,e.Empname,e.Deptno,d.Deptno,d.Deptname

(e(Employee) X e(Department))

e.E mpno	e.Empname	e.Deptno	d.Deptno	d.Deptname
101	Ramesh	100	100	Sales
101	Ramesh	100	200	R &D
102	Ramesh	200	100	Sales
102	Ramesh	200	200	R &D
103	Rajesh	101	100	Sales
103	Rajesh	101	200	R &D

9.3.1 Join Operator 

Join operator is used to retrieve data from multiple table or relations

Syntax

<tablename>  <tablename>

There are various types of join in relational algebra

Natural Joins

- Assume R1 and R2 have attributes A in common. Natural join is formed by concatenating all tuples from R1 and R2 with same values for A, and dropping the occurrences of A in R2
- $R1 \bowtie R2 = \pi_{A'}(\sigma_C(R1 \times R2))$
- where C is the condition that the values for R1 and R2 are the same for all attributes in A and A' is all attributes in R1 and R2 apart from the occurrences of A in R2.

Course Table

CourseId	Title	eid
CS51T	DBMS	123
CS52S	OS	345
CS52T	Networking	345
CS51S	ES	456

Instructor Table

eid	ename
123	Rao
345	Allen
456	Mansingh

Course \bowtie Instructor

CourseId	Title	eid	ename
CS51T	DBMS	123	Rao
CS52S	OS	345	Allen
CS52T	Networking	345	Allen
CS51S	ES	456	Mansingh

CourseId,ename Course \bowtie Instructor

CourseId	ename
CS51T	Rao
CS52S	Allen
CS51S	Mansingh

8.3.2 Inner Join

- In Inner join, tables are joined together where there is the match (=) of the primary and foreign keys.

$R \bowtie \langle R.\text{primary_key} = S.\text{foreign_key} \rangle S$

Inner joins return rows only when there is at least one row from both tables that matches the join condition. Inner joins eliminate the rows that do not match with a row from the other table

Student Table

Studid	name	course
100	Jayesh	PH
200	Preetesh	CM
300	Pramod	CM

Course Table


course#	name
PH	Pharmacy
CM	Computing

Students \bowtie course = course# Courses

Studid	name	course	course#	Course.name
100	Jayesh	PH	PH	Pharmacy
200	Preetesh	CM	CM	Computing
300	Pramod	CM	CM	Computing

8.3.3 Outer Join

- Inner join + rows of one table which do not satisfy the condition.
- Left Outer Join:

- R  $\langle R.primary_key = S.foreign_key \rangle$ S
- All rows from R are retained and unmatched rows of S are padded with NULL

Student Table

Studid	name	Course#
100	Jayesh	PH
200	Preetesh	CM
400	Pramod	EN

Course Table


course#	Cname
PH	Pharmacy
CM	Computing
CH	Chemistry

e.studid,e.name,e.Course#,c.Course#,c.Cname

(e (Student)= ∞ c (Course)

e.studid	e.name	e.course#	c.course#	c.Cname
100	Jayesh	PH	PH	Pharmacy
200	Preetesh	CM	CM	Computing
400	Pramod	EN	NULL	NULL

8.3.4 Right Outer Join

- Right Outer Join: R  $\langle R.primary_key = S.foreign_key \rangle$ S

All rows from S are retained and unmatched rows of R are padded with NULL

Right outer Join takes all the record form the right relation S that unmatched any record in the S relation

Student Table

Studid	name	Course#
100	Jayesh	PH
200	Preetesh	CM
400	Pramod	EN

Course Table

course#	Cname
PH	Pharmacy
CM	Computing
CH	Chemisty

e.studid,e.name,e.Course#,c.Course#,c.Cname

(e (Student) \bowtie c (Course))

e.studid	e.name	e.course#	c.course#	c.Cname
100	Jayesh	PH	PH	Pharmacy
200	Preetesh	CM	CM	Computing
NULL	NULL	NULL	CH	Chemisty

8.3.5 Full Outer Join

In Full outer join tables on the both sides of operator contains null values

It will **contain record from both relations** that do not join with any record from the other relation. Those tuples will be padded with NULLs as usual.

R COLA	R COLB
A	1
B	2
D	3
F	4
E	5

S COLA	S COLB
A	1
C	2
D	3
E	4

R.ColA = S.SColA

A	1	A	1
D	3	D	3
E	5	E	4
B	2	NULL	NULL
F	4	NULL	NULL
NULL	NULL	C	2

8.3.5 Relational Division Operator

- It is denoted as \div .

Let $r(R)$ and $s(S)$ be relations

$r \div s$: - the result consists of the restrictions of tuples in r to the attribute names unique to R , i.e. in the Header of r but not in the Header of s , for which it holds that all their combinations with tuples in s are present in r .

Relation or table "r":-

A	B
a	1
b	2
a	2
p	3
p	4

Relation or table "s":-

B
2
3

Therefore $r \div s$

A
b
a
p

8.4 EXTENDED RELATIONAL OPERATOR

8.4.1 Duplicate-elimination operator

This operator is used to remove the duplicate record from the relation

Duplicate-elimination operator is denoted by

$(R)=$ indicates that relation with one copy of each tuple that appears one or more times in R

Example

R=

A	B
1	2
3	4
1	2

$(R)=$

A	B
1	2
3	4

Sorting

$L(R)$ = list of tuples of R, ordered according to attributes on list L

cannot be followed by other relational operators.

Example

R=

A	B
1	3
3	4
5	2

$B(R) = [(5,2), (1,3), (3,4)]$

9.4.2 Aggregation Operators

Operators that summarise or aggregate the values in a single attribute of a relation.

Operators are the same in relational algebra and SQL.

All operators treat a relation as a bag of tuples.

SUM: computes the sum of a column with numerical values.

AVG: computes the average of a column with numerical values.

MIN and MAX: for a column with numerical values, computes the smallest or largest value, respectively. for a column with string or character values, computes the lexicographically smallest or largest values, respectively.

COUNT: computes the number of non-NULL tuples in a column.

In SQL, can use COUNT (*) to count the number of tuples in a relation.

Grouping operator

$\sigma_L(R)$ where L is a list of items in the Relation(R) that are either

- They are individual attributes or grouping attributes or
- (A) , Where σ is an aggregation operator and A the attribute in the relation(R) to which the aggregation operator is to be applied

It is computed by:

- Group R according to all the grouping attributes on list L.
- Within each group, compute (A) , for each element (A) on list L.
- Result is the relation whose columns consist of one tuple for each group. The components of that tuple are the values associated with each element of L for that group.

Example

Let R =

Mall	Jeans	Price
R-Mall	Killer	1500
Metro Mall	Lee	1700
Phoenix Mall	Live's	1800
<u>INORBIT MALL</u>	Killer	1900
Spykar	Lee	1400

Compute $\sigma_{\text{Jeans,AVG(Price)}}$

Group by the grouping attribute(s), Jeans in this case:

Mall	Jeans	Price
R-Mall	Killer	1500
<u>INORBIT MALL</u>	Killer	1900
Metro Mall	Lee	1700
Spykar	Lee	1400
Phoenix Mall	Live's	1800

Compute average of price within groups:

Jeans	Price
Killer	3400
Lee	3100
Live's	1800



RELATIONAL CALCULUS

Unit Structure

9.0 Objectives

Relation calculus comes from one of the mathematical branches or logic is called predicate calculus.

The differentiate between the relational algebra and relation calculus : relational algebra provides a series of procedures that is used for solving the problem and relational algebra is describe what is problem is

It is closer than relational algebra to how users would formulate queries in terms of their information needs, rather than in terms of operations.

Relational calculus is categerious in to two part

- 1) Tuple relational calculus
- 2) Domain relational Calculus

Relational Calculus is an non prodeural language where as relational algebra is procedural language

9.1 TUPLE RELATIONAL CALCULUS

Tuple calculus is a calculus that was introduced by Edgar F. Codd as part of the relational model, in order to provide a declarative database-query language for this data model.

Tuple relational calculus is a non procedural language

We must provide a formal description of the information desired.

Each queries in the Tuple Relation calculus is written as

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

i.e It is set of tuples t for which predicate P is true

We can also use the notation for describing the tuple calculus

We use $t[a]$ to indicate the value of tuple t on attribute a

We use $t \in r$ to indicate that tuple t is in relation r

9.1.1 Selection and Projection

To find out the data from the table we have to use the operator known as selection and projection that used to select desired data by applying some predicate calculus or formula on table

In Tuple relational Calculus a query can be written for selection operation() as

$$p(r) = \{t \mid P(t)\}$$

Where as

$$p(r) = \text{Selection operator on the Relation}$$

t = Set of tuples(as called as variable range over tuples)

p = Predicate indicate that is true for t

each formula in the relation calculus is consist of Connectivity by logical operator such as

(\wedge), or (\vee), not (\neg)

Set of comparison operators: (e.g., $<$, \leq , $=$, \neq , $>$, \geq)

Implication (\Rightarrow): $x \Rightarrow y$, if x is true, then y is true

$$x \Rightarrow y \equiv \neg x \vee y$$

Set of quantifiers:

- $\exists t \in r (Q(t)) \equiv$ "there exists" a tuple in t in relation r such that predicate $Q(t)$ is true
- $\forall t \in r (Q(t)) \equiv Q$ is true "for all" tuples t in relation r

Query to select all attributes of the table

Consider a sample database of an Employee

SSN	FirstName	LastName	Salary
101	Jayesh	Shinde	30000
102	Preetesh	Shinde	40000
103	Sachin	Tendulkar	50000
104	Pravin	Kanetkar	35000
105	Mahesh	Jadhav	53000

We Want to find all the record of employee table using relational calculus

Select all the Employee whose having the salary more than 30000

$$\text{salary} > 30000 (\text{Employee}) = \{t | t \in \text{Employee} \wedge t[\text{Salary}] > 30000\}$$

SSN	FirstName	LastName	Salary
102	Preetesh	Shinde	40000
103	Sachin	Tendulkar	50000
104	Pravin	Kanetkar	35000
105	Mahesh	Jadhav	53000

Query 2 Find the SSN for each Employee whose having salary more than 30000

$$\text{SSN} (\text{Salary} > 30000 (\text{Employee})) = \{t | \exists t \in \text{Employee} [\text{SSN}] = t[\text{SSN}] \wedge t[\text{Salary}] > 30000\}$$

SSN
102
103
104
105

9.1.2 SET Operations

In set operation, two or more select statement is combined together to form as desired result

On other hand the set operation combines rows from two or more different queries

In select statement, there must be same number of columns retrieve from the two or more queries

There must be same data type or compatible type of each columns in select statement

In tuple relational calculus ,a query can be written as

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

Where,

t= Set of tuples(as called as variable range over tuples)

p= Predicate indicate that is true for t

Reserves Table

SID	BID	Day
22	101	10/10/96
95	103	11/12/96

Sailor Table

SID	Sname	rating	age
22	Jayesh	7	45.0
31	Preetesh	8	55.5
95	Pramod	3	63.5

Find the all Sailors ID whose rating is greater than 2 ,here we use the union operator in the relational algebra,In the relational calculus we used two exists clause and Connected by or

$$\pi_{SID}(\text{Reserves}) \cup \pi_{SID}(\text{Sailor}) = \{t | \exists s \in \text{Reserves}(t[SID]=s[SID]) \vee \exists u \in \text{Sailor}(t[SID]=u[SID] \wedge \text{rating} > 2)\}$$

SID
22
31
95

In This above Example duplicate record is eliminated

Query to select the data using Intersection operator

Find the those Sailors ID whose rating is greater than 7 ,here we use the Intersect operator in the relational algebra,In the relational calculus we used two exists clause and Connected by And

$$\pi_{SID} (\text{Reserves}) \cap \pi_{SID} (\text{Sailor}) = \{t \mid \exists s \in \text{Reserves}(t[SID]=s[SID]) \wedge \exists u \in \text{Sailor}(t[SID]=u[SID] \wedge \text{rating} > 7)\}$$

SID
31

In This above Example duplicate record is eliminated

Query to select data using difference operator

9.1.3 Cartesian Product Operation

In Cartesian product operation defines as every tuples of relation R combines with every relation S

In the Relational Cartesian Product , The result will return as all the attributes from both relation R and S.

Syntax

$$r \times s = \{t \in q \mid t \in r \text{ and } t \in s\}$$

Consider two table Employee and Department

EmployeeID	Designation
101	Lecturer
102	Assistant Professor
103	Professor

DepartNumber	DepartName
E1	Electrical
C1	Computer
E3	Electronics

In The tuple Relational calculus , requires two exists clause they are connected by \wedge

The Query Can be Written as

$$\text{EmployeeID} \quad (\text{EmployeeXDepartment}) = \{t \mid \exists s \in \text{Employee} \wedge \exists u \in \text{Department}\}$$

EmployeeID	Designation	DepartNumber	DepartName
101	Lecturer	E1	Electrical
102	Assistant Professor	E1	Computer
103	Professor	E1	Electronics
101	Lecturer	C1	Electrical
102	Assistant Professor	C1	Computer
103	Professor	C1	Electronics
101	Lecturer	E3	Electrical
102	Assistant Professor	E3	Computer
103	Professor	E3	Electronics

9.1.4 Join Operator

Join operator is used to retrieves data from mutiple relations

Syntax

$$r \bowtie s = \{t \in q \mid t \in r \text{ and } t \in s\}$$

Example

Retrives data from the two table knowns as Employee and Department

EmployeeID	Designation	DepartNumber
101	Lecturer	E1
102	Assistant Professor	C1
103	Professor	E3

DepartNumber	DepartName
E1	Electrical
C1	Computer
E3	Electronics

In The tuple Relational calculus , requires two exits clause they are connected by \wedge

Find the Employee Id whose teaches in the Computer Department

$$\text{EmployeeID} \quad (\text{Employee} \bowtie \text{Department}) = \{t \mid \exists s \in \text{Employee} (t[\text{EmployeeID}] = s[\text{EmployeeID}]) \wedge \exists$$

$u \in \text{Department}(u[\text{DepartNumber}] = s[\text{DepartNumber}] \wedge u[\text{DepartName}] = \text{'Computer'})$

EmployeeID
102

9.1.5 Division Operator

The division of relation R over relation S is denoted by $R \div S$

Consider an example Student table has two attributes Student Name and Marks and another table is Marks

Student name	Marks
Dinesh	97
Arun	100
Kamal	98
Jay	85
Virat	98
Mahendra	95
Dharmendra	95

Marks
98

10. 2 Domain Relational Calculus

In computer science, **domain relational calculus (DRC)** is a calculus that was introduced by Michel Lacroix and Alain Pirotte as a declarative database query language for the relational data model.

In DRC, *queries* have the form:

$$\{ \langle X_1, X_2, \dots, X_n \rangle \mid p(\langle X_1, X_2, \dots, X_n \rangle) \}$$

where each X_i is either a domain variable or constant, and $p(\langle X_1, X_2, \dots, X_n \rangle)$ denotes a DRC *formula*. The result of the query is the set of tuples X_i to X_n which makes the DRC formula true.

This language uses the same operators as tuple calculus, the logical connectives \wedge (and), \vee (or) and \neg (not). The existential

quantifier (\exists) and the universal quantifier (\forall) can be used to bind the variables.

Its computational expressiveness is equivalent to that of Relational algebra

Example - Domain Relational Calculus

1) Find the names of all Clerks who earn more than RS 10,000.

{fN, IN | (sN, posn, sex, DOB, sal, bN) (Staff (sN, fN, IN, posn, sex, DOB, sal, bN)

posn = 'Clerks'sal > 10,000)}

2) List the staff who manage properties for rent in Mumbai.

{sN, fN, IN, posn, sex, DOB, sal, bN | (sN1,cty)
(Staff(sN,fN,IN,posn,sex,DOB,sal,bN) PropertyForRent(pN, st, cty,
pc, typ, rms, rnt, oN, sN1, bN1) (sN=sN1) cty='Mumbai')}

3) List the names of staff who currently do not manage any properties for rent.

{fN, IN | (sN) (Staff(sN,fN,IN,posn,sex,DOB,sal,bN) (~(sN1)
(PropertyForRent(pN, st, cty, pc, typ, rms, rnt, oN, sN1,
bN1)(sN=sN1))))}

4) Retrieve names of all professors who have taught Management345

{ N \exists I professor.Id D Professor.Dept.Id (Professor
(I,N,D) AND S Teaching .Semester (Teaching
(I,MGT345,S)))}

This can be abbreviated

{N | Professor (I, N, D) AND(Teaching(I,MGT345,S))}

1) All courses that have been taken by every student:

{ C | Course(D,C,C,D) AND S Students .Id (Transcript
(S,C,SEM,G))}

2) Find all students who have ever taken a course from every professor who has ever taught a course.

{I | Transcript(S,C,SEM,G1) AND PI Teaching .ProfId
(Teaching(PI,C2,SEM2) AND Transcript(S,C,SEM,G2)

- 3) Retrieve IDs of students who did not take any courses in F2001:

$$\{l \mid \text{Student}(l, N, A, S) \text{ AND NOT Transcript}(l, C, F2001, G)\}$$

- 4) Find potential student graders for this semester's courses:

$$\{P, C, S \mid \text{Teaching}(P, C, S2002) \text{ AND Transcript}(S, C, SEM, G) \text{ AND SEM} \neq S2002\}$$

- 5) Find all loan numbers for loans with an amount greater than \$1200:

$$\{ \langle l \rangle \mid \exists a, b (\langle l, a, b \rangle \in \text{loan} \wedge a > 1200) \}$$

Equivalent Relational Algebra expression

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

- 10) Find the loan numbers of all loans made jointly to Amit and Ramesh.

$$\{ \langle l \rangle \mid \exists x (\langle x, l \rangle \in \text{borrower} \wedge x = \text{"Amit"}) \wedge \exists x (\langle x, l \rangle \in \text{borrower} \wedge x = \text{"Ramesh"}) \}$$

- 11) Find the names of all customers who have a loan from the Kurla branch, and find the loan amount.

$$\{ \langle c, a \rangle \mid \exists l (\langle c, l \rangle \in \text{borrower} \wedge \exists b (\langle l, b, a \rangle \in \text{loan} \wedge b = \text{"Kurla"}) \}$$

- 12) Find branch name, loan number, customer name and amount for loans of over \$1200.

$$\{ \langle b, l, c, a \rangle \mid \langle b, l, c, a \rangle \in \text{borrow} \wedge a > 1200 \}$$

- 13) Find all customers who have a loan for an amount > than \$1200.

$$\{ \langle c \rangle \mid \exists b, l, a (\langle b, l, c, a \rangle \in \text{borrow} \wedge a > 1200) \}$$

- 14) Find all customers having a loan from the MTU branch, and the city in which they live.

$$\{ \langle c, x \rangle \mid \exists b, l, a (\langle b, l, c, a \rangle \in \text{borrow} \wedge b = \text{"MTU"} \wedge \exists y (\langle y, x \rangle \in \text{Customer})) \}$$

15) Find all customers having a loan, an account or both at the MTU branch.

$$\{ \langle c, x \rangle \mid \exists b, l, a (\langle b, l, c, a \rangle \text{ borrow } \wedge b = \text{"MTU"} \vee \langle b, a, c, n \rangle \text{ deposit } \wedge b = \text{"MTU"}) \}$$

16) Find all customers who have an account at all branches located in Kurla.

$$\{ \langle c \rangle \mid \forall x, y, z (\neg (\langle x, y, z \rangle \text{ branch } \wedge z = \text{"Kurla"} \wedge \neg \exists a, n (\langle x, a, c, n \rangle \text{ deposit }))) \}$$

9.3 RELATIONAL ALGEBRA VS RELATIONAL CALCULUS

Sr.No	Relational Algebra	Relational Calculus
1	Relational Algebra is a procedural query language, very useful for representing execution plans, relatively close to SQL.	The tuple Relational calculus is a non-procedural language, Lets users describe what they want, rather than how to compute it.
2	Relational algebra indicates operation on table that produces a new tables as a result	Relation Calculus defines a new table by providing representation in term of given relation
3	In relation algebra , A query can be written with help of relational operator known as selection, projection etc. $\text{ColumnName}(\text{Condition}(\langle \text{TableName} \rangle))$ Table is name of the input relation	In Relational Calculus A query Can be written as $\{t \mid P(t)\}$ I.e The set of tuple t where Predicate P is true
4	In relation algebra , we need provide a series of procedures that use to generated the answer to respond of set of query	In relational Calculus we needs to provides a formal description of the information



UNIT V: CONSTRAINTS, VIEWS AND SQL

10

CONSTRAINTS

Unit Structure

- 10.0 Objectives
- 10.1 Introduction
- 10.2 Types of Constraints
- 10.3 Integrity Constraints

10.0 OBJECTIVES:

- What are constraints?
- What are types of constraints?
- Integrity constraints

10.1 INTRODUCTION

Definition:

Constraints are used to limit the type of data that can go into a table.

s

Constraints can be specified when a table is created (with the CREATE TABLE statement) or after the table is created (with the ALTER TABLE statement).

Syntax:

```
    Create table table_name
{
    Column
data_type[column_constraint_Name][Column_constraint],
    Column datatype[DEFAULT expr] [column_constraint],
    .....
    [table_constraint][....]
}
```

Example:

Some attributes in the table are not required so such columns can be defined as NULL constraint. In the EMPLOYEE table it is allowed insert row having Phone number column as NULL.

```

Create table EMPLOYEE
{
    Did varchar(10),
    EName varchar(10),
    Phone_Number char (100) NULL
}

```

Data Integrity

Constraints are used to enforce the data integrity. This ensures the accuracy and reliability of the data in the database. The following categories of the data integrity exist:

- Entity Integrity
- Domain Integrity
- Referential integrity
- User-Defined Integrity

Entity Integrity ensures that there are no duplicate rows in a table.

Ex: Unique, Primary Key

Domain Integrity enforces valid entries for a given column by restricting the type, the format, or the range of possible values.

Ex: check, Null, not Null

Referential integrity ensures that rows cannot be deleted, which are used by other records (for example, corresponding data values between tables will be vital).

Ex: Foreign Key

User-Defined Integrity enforces some specific business rules that do not fall into entity, domain, or referential integrity categories.

10.2 TYPES OF CONSTRAINTS

Constraints can be defined in **two** ways:-

1) The constraints can be specified immediately after the column definition. This is called column-level definition.

2) The constraints can be specified after all the columns are defined. This is called table-level definition.

Some of the Constraints are listed below:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

NOT NULL CONSTRAINT

- The NOT NULL constraint enforces a column to NOT accept NULL values.
- The NOT NULL constraint enforces a field to always contain a value. This means that you cannot insert a new record, or update a record without adding a value to this field.

Syntax to define a Not Null constraint:

[CONSTRAINT constraint name] NOT NULL

For Example:

To create an employee table that enforces the "E_Id" column and the "LastName" column to not accept NULL values:

```
Create Table EMPLOYEE
(
  E_Id int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255)
)
```

➤ UNIQUE KEY CONSTRAINT

This constraint ensures that a column or a group of columns in each row have a distinct value. A column(s) can have a null value but the values cannot be duplicated.

Syntax to define a Unique key at column level:

[CONSTRAINT constraint_name] UNIQUE

Syntax to define a Unique key at table level:

[CONSTRAINT constraint_name] UNIQUE(column_name)

For Example: To create an employee table with Unique key, the query would be like,

Unique Key at column level:

<pre>CREATE TABLE employee (E_Id number(5) PRIMARY KEY, name char(20), dept char(10), age number(2), salary number(10), location char(10) UNIQUE);</pre>	<pre>CREATE TABLE employee (E_Id number(5) PRIMARY KEY, name char(20), dept char(10), age number(2), salary number(10), location char(10) CONSTRAINT loc_un UNIQUE);</pre>
OR	

Unique Key at table level:

```
CREATE TABLE employee
(E_Id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10),
CONSTRAINT loc_un UNIQUE(location)
);
```

➤ PRIMARY KEY CONSTRAINTS:

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain unique values.

A primary key column cannot contain NULL values.

Each table should have a primary key, and each table can have only ONE primary key.

Syntax to define a Primary key at column level:

```
column name datatype [CONSTRAINT constraint_name] PRIMARY
KEY
```

Syntax to define a Primary key at table level:

```
[CONSTRAINT constraint_name] PRIMARY KEY
(column_name1,column_name2,..)
```

- **column_name1, column_name2** are the names of the columns which define the primary Key.
- The syntax within the bracket i.e. [CONSTRAINT constraint_name] is optional.

For Example: To create an employee table with Primary Key constraint, the query would be like.

Primary Key at table level:

<pre>CREATE TABLE employee (E_Id number(5) PRIMARY KEY, name char(20), dept char(10), age number(2), salary number(10), location char(10));</pre>	<pre>CREATE TABLE employee (E_Id number(5) CONSTRAINT emp_id_pk PRIMARY KEY, name char(20), dept char(10), age number(2), salary number(10), location char(10));</pre>
OR	

Primary Key at table level:

```
CREATE TABLE employee
(E_Id number(5),
name char(20),
dept char(10),
age number(2),
salary number(10),
location char(10),
CONSTRAINT emp_id_pk PRIMARY KEY (id)
);
```

➤ **FOREIGN KEY CONSTRAINT**

- This constraint identifies any column referencing the PRIMARY KEY in another table.
- It establishes a relationship between two columns in the same table or between different tables.
- For a column to be defined as a Foreign Key, it should be a defined as a Primary Key in the table which it is referring. One or more columns can be defined as foreign key.

Syntax to define a Foreign key at column level:

```
[CONSTRAINT constraint_name] REFERENCES
Referenced_Table_name(column_name)
```

Syntax to define a Foreign key at table level:

```
[CONSTRAINT constraint_name] FOREIGN KEY(column_name)
REFERENCES referenced_table_name(column_name);
```

For Example:

1) Lets use the "product" table and "order_items".

Foreign Key at column level:

<pre>CREATE TABLE product (product_id number(5) CONSTRAINT pd_id_pk PRIMARY KEY, product_name char(20), supplier_name char(20), unit_price number(10));</pre> <p>OR</p>	<pre>CREATE TABLE order_items (order_id number(5) CONSTRAINT od_id_pk PRIMARY KEY, product_id number(5) CONSTRAINT pd_id_fk REFERENCES product(product_id), product_name char(20), supplier_name char(20), unit_price number(10));</pre>
--	--

Foreign Key at table level:

```
CREATE TABLE order_items
( order_id number(5) ,
product_id number(5),
product_name char(20),
supplier_name char(20),
unit_price number(10)
CONSTRAINT od_id_pk PRIMARY KEY(order_id),
CONSTRAINT pd_id_fk FOREIGN KEY(product_id)
REFERENCES product(product_id)
);
```

2) If the employee table has a 'mgr_id' i.e, manager id as a foreign key which references primary key 'id' within the same table, the query would be like,

```
CREATE TABLE employee
(E_Id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
mgr_id number(5) REFERENCES employee(id),
salary number(10),
location char(10)
);
```

➤ **CHECK CONSTRAINTS:**

This constraint defines a business rule on a column. All the rows must satisfy this rule. The constraint can be applied for a single column or a group of columns.

Syntax to define a Check constraint:

```
[CONSTRAINT constraint_name] CHECK (condition)
```

For Example: In the employee table to select the gender of a person, the query would be like

Check Constraint at column level:

```
CREATE TABLE employee
(E_Id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
gender char(1) CHECK (gender in ('M','F')),
salary number(10),
location char(10)
);
```

Check Constraint at table level:

```
CREATE TABLE employee
(E_Id number(5) PRIMARY KEY,
name char(20),
dept char(10),
age number(2),
gender char(1),
salary number(10),
location char(10),
CONSTRAINT gender_ck CHECK (gender in ('M','F'))
);
```

10.3 INTEGRITY CONSTRAINTS

- Constraints are used to maintain integrity of database so they are also called as data Integrity constraints
- Data integrity constraints provide a way of ensuring that changes made to the database by authorised users do not result in a loss of data consistency and correctness.
- An integrity constraint can be any arbitrary predicate or condition applied to the database.
- Integrity constraints may be difficult to evaluate, so will only consider integrity constraints that can be tested easily with minimal overhead.
- Integrity constraint with **E-R** models
- **Key declarations:** ability that the certain attributes of relations can form a candidate key for a given entity set.
- **Form of a relationship:** Mapping cardinalities like 1:1, 1-Many and Many to many.
- To maintain integrity in the database we have many types of constraints which can keep database in integrity state.



VIEWS

Unit Structure

11.0 Objectives

11.1 Introduction

11.0 OBJECTIVES:

- Introduction to views
- Data Independence
- Security
- Updates on views
- Comparison between tables and views

11.1 INTRODUCTION

Definition:

- A view is a virtual table that consists of columns from one or more tables.
- A virtual table is like a table containing fields but it does not contain any data. In run time it contains the data and after that it gets free.
- But table stores the data in database occupy some space.
- Just like table, view contains Rows and Columns which is fully virtual based table.
- **Base Table** -The table on which view is defined is called as Base table.

CREATING A VIEW

This statement is used to create a view.

Syntax:

```
CREATE VIEW view_name
```

- The CREATE statement assigns a name to the view and also gives the query which defines the view.
- To create the view one should must have privileges to access all of the base tables on which view is defined.
- The create view can change name of column in view as per requirements.

HORIZONTAL VIEW

A Horizontal view will restrict the user's access to only a few rows of the table.

Example:

Define a view for Sue (employee number 1004) containing only orders placed by customers assigned to her.

```
CREATE VIEW SUEORDERS AS
SELECT *
FROM ORDERS
WHERE CUST IN
(SELECT CUST_NUM FROM CUSTOMERS WHERE
CUST_REP=1004)
```

VERTICAL VIEW

A vertical view restricts a user's access to only certain columns of a table.

Ex:

```
CREATE VIEW EMP_ADDRESS AS
SELECT EMPNO, NAME, ADDR1, ADDR2, CITY
FROM EMPLOYEE
```

ROW/COLUMN SUBSET VIEW.

- Views can be used to restrict a user to access only selected set of rows and columns of a table in a database.
- This view generally helps us to visualize how view can represent the base table.
- This type of view is combination of both horizontal and vertical views.

Ex:

```
CREATE VIEW STUDENTS_PASSED AS
SELECT ROLLNO, NAME, PERCENTAGE
FROM STUDENTS
WHERE RESULT ='PASS'
```

GROUPED VIEW

- A grouped view is one in which query includes GROUPBY CLAUSE.
- It is used to group related rows of data and produce only one result row for each group.

Ex:

Find summary information of Employee Salaries in sales Department.

Defining View

```
CREATE VIEW Summary_Empl_Sal
(
Total_Employees,
Minimum_salary,
Maximum_Salary,
Average_salary,
Total_salary
)
AS
SELECT COUNT(EmpID),
Min(Salary),
Max(Salary),
Avg(Salary),
SUM(Salary),
FROM Employee
GROUP BY Department
HAVING Department='Sales';
```

View Call

```
Selelct *
From Summary_Empl_Sal
```

The above Query will give,
Total No. Of Employees in sales Department, Minimum Salary in sales Department.

Maximum Salary in sales Department.
 Average Salary in sales Department.
 Total Salary of Employees in sales Department.

JOINED VIEWS

- A Query based on more than one base table is called as Joined View.
- It is also called as Complex View
- This gives a way to simplify multi table queries by joining two or more table query in the view definition that draws its data from multiple tables and presents the query results as a single view.
- The view once it is ready we can retrieve data from multiple tables without joining any table simply by accessing a view created.

Ex:

Company database find out all EMPLOYEES for respective DEPARTMENTS.

Schema Definition:

EMPLOYEE-> EmpID, EmpName, Salary, DeptID
 DEPARTMENT-> DeptID, DeptName

View Definition

```
CREATE VIEW Emp_Details
As
Select Employee,EmpID,
Department, DeptID,
Department, DeptName
From
Where Employee.DeptID=Department.DeptID;
```

View Call

```
Select * from Emp_Details
```

DROPPING VIEW

When a view is no longer needed, it can be removed by using DROP VIEW statement.

Syntax:

```
DROP VIEW <VIEW NAME> [CASCADE/RESTRICT]
```

CASCADE: It deletes the view with all dependent view on original view.

RESTRICT: It deletes the view only if they're in no other view depends on this view.

Example:

Consider that we have view VABC and VPQR .View VPQR depends on VABC.

Query:

DROP view VABC

If we drop VABC, then cascading affect takes place and view VPQR is also dropped.

Thus default option for dropping a view is CASCADE. The CASCADE option tells DBMS to delete not only the named view, but also query views that depend on its definition. But,

QUERY:

DROP view VABC RESTRICT

Here, the query will fail because of RESTRICT option tells DBMS to remove the view only if no other views depend on it. Since VPQR depends on VABC, will cause an error.

UPDATING VIEWS

- Records can be updated, inserted, and deleted though views.
- UPDATAEBLE VIEWS are those in which views are used against INSERT, DELETE and UPDATE statements.

The following conditions must be fulfilled for view updates:

- DISTINCT must not be specified; that is, duplicate rows must not be eliminated from the query results.
- The FROM clause must specify only one updateable table; that is, the view must have a Single source table for which the user has the required privileges. If the source table is itself a view, then that view must meet these criteria.
- Each select item must be a simple column reference; the select list cannot contain expressions, calculated columns, or column functions.
- The WHERE clause must not include a subquery; only simple row-by-row search conditions may appear.
- The query must not include a GROUP BY or a HAVING clause.

Data Independence

A major purpose of a database system is to provide the users with an abstract view of data.

To hide the complexity from users database apply different levels of abstraction. The following are different levels of abstraction.

- i. Physical Level
- ii. Logical Level
- iii. View Level

Physical Level

- Physical Level is the lowest level of abstraction and it defines the storage structure.
- The physical level describes complex low level data structures in detail.
- The database system hides many of the lowest level storage details from the database programmers.
- Database Administrators may be aware of certain details of physical organization of data.

Logical Level

- This is the next higher level of abstraction which describe what data are stored in database, relation between data, types of data etc .
- Database programmers, DBA etc knows the logical structure of data

View Level

- This the highest level of abstraction.
- It provides different view to different users. At the view level users see a set of application programs that hide details of data types.
- The details such as data type etc are not available at this level.
- Only view or Access is given to a part of data according to the users access right

Physical Data Independence

The changes in Physical Level does not affect or visible at the logical level.

This is called physical data independence.

Logical Data Independence

The changes in the logical level do not affect the view level. This is called logical data independence.

ADVANTAGES OF VIEWS

1. **Security** Each user can be given permission to access the database only through a small set of views that contain the specific data the user is authorized to see, thus restricting the user's access to stored data.

2. **Query simplicity** A view can draw data from several different tables and present it as a single table, turning multi-table queries into single-table queries against the view.

3. **Structural simplicity** Views can give a user a personalized view of the database structure, presenting the database as a set of virtual tables that make sense for that user.

4. **Insulation from change** A view can present a consistent, unchanged image of the structure of the database, even if the underlying source tables are split, restructured, or renamed. Note, however, that the view definition must be updated whenever underlying tables or columns referenced by the view are renamed.

5. **Data integrity** If data is accessed and entered through a view, the DBMS can automatically check the data to ensure that it meets specified integrity constraints.

DISADVANTAGES OF VIEWS

While views provide substantial advantages, there are also three major disadvantages to using a view instead of a real table:

- **Performance**

Views create the appearance of a table, but the DBMS must still translate queries against the view into queries against the underlying source tables.

If the view is defined by a complex multitable query, then even a simple query against the view becomes a complicated join, and it may take a long time to complete.

However, the issue isn't because the query is in a view—any poorly constructed query can present performance problems—the hazard is that the complexity is hidden in the view, and thus users are not aware of how much work the query is performing.

- **Manageability**

Like all database objects, views must be managed. If developers and database users are allowed to freely create views without controls or standards, the DBA's job becomes that much more difficult.

This is especially true when views are created that reference other views, which in turn reference even more views.

The more layers between the base tables and the views, the more difficult it is to resolve problems attributed to the views.

- **Update restrictions**

When a user tries to update rows of a view, the DBMS must translate the request into an update on rows of the underlying source tables.

This is possible for simple views, but more complex views cannot be updated; they are read-only.

COMPARISON BETWEEN TABLES AND VIEWS

VIEWS

- View comprises of Query in view definition.
- Just like table, view contains Rows and columns which is fully virtual based table.
- The fields in a view are fields from one or more real tables in the database.
- When view is called, it does not contain any data. For that, it goes to memory and fetches data from base table and displays it.
- E-g: - An I.T. Faculty requires only I.T. related data of students so we can create view called as **Stud_IT_View** for Faculty as below which will only depicts I.T. data of students to I.T. faculty.
- A virtual table is like a table containing fields but it does not contain any data. In run time it contains the data and after that it gets free. But table stores the data in database occupy some space.

Stud_IT_View (Student_Id, Student_Name, I.T.)

We can also add functions like WHERE and JOIN statements to a view and present the data as if the data were coming from one single table.

TABLES

- Table stores the data and database occupies some space in database.
- Tables contain rows and columns, columns representing fields and rows containing data or records.

EX:

Consider a Employee containing following columns,
EMPLOYEE (Emp_ID, EmpName, Designation, Address, Salary)



STRUCTURED QUERY LANGUAGE

Unit Structure

12.0 Objectives

12.1 Introduction

12.0 OBJECTIVES:

- Data Definition
- Aggregate Functions
- Null Values
- Nested Sub queries
- Joined relations
- Triggers

12.1 INTRODUCTION

- SQL stands for Structured Query Language
- It lets you access and manipulate databases.
- SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s.
- The first commercial relational database was released by Relational Software (Later called as Oracle).
- SQL is not a case sensitive as it is a keyword based language and each statement begins with a unique keyword.

FEATURES OF SQL

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert ,Update, Delete, records in a database
- SQL can create stored procedures in a database
- SQL can create views in a database

SQL COMMANDS:

- SQL commands are instructions used to communicate with the database to perform specific task that work with data.
- SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can create tables, add data to tables, or modify data, drop the table, set permissions for users.
- SQL commands are grouped into 2 major categories depending on their functionality:

Data Definition Language (DDL) - These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are CREATE, ALTER, DROP, RENAME, and TRUNCATE.

Data Manipulation Language (DML) - These SQL commands are used for storing, retrieving, modifying, and deleting data. These commands are SELECT, INSERT, UPDATE, and DELETE.

DATA DEFINITION LANGUAGE (DDL)

- DDL statements are used to build and modify the objects and structure of tables in database.
- The DDL part of SQL permits database tables to be created or deleted.
- It also defines indexes (keys), specifies links between tables, and imposes constraints between tables.
- The most important DDL statements in SQL are:
 - **CREATE TABLE** - creates a new table
 - **ALTER TABLE** - modifies a table
 - **DROP TABLE** - deletes a table
 - **CREATE INDEX** - creates an index (search key)
 - **DROP INDEX** - deletes an index

a. CREATE COMMAND

This statement used to create Database.

Syntax:

```
CREATE TABLE tablename
(
    column_name data_type attributes...,
    column_name data_type attributes...,
    ...
)
```

- Table and column names can't have spaces or be "reserved words" like TABLE, CREATE, etc.

Example:

```
CREATE TABLE Employee
(
    EmpId varchar2(10),
    FirstName char(20),
    LastName char(20),
    Designation char(20),
    City char(20)
);
```

OUTPUT:

Emp_Id	FirstName	LastName	Designation	City
--------	-----------	----------	-------------	------

b. ALTER COMMAND:

- This statement is used to make modifications to the table structure.
- This statement is also used to add, delete, or modify columns in an existing table

Syntax:

```
ALTER TABLE table_name
ADD column_name datatype
```

OR

```
ALTER TABLE table_name
DROP COLUMN column_name
```

OR

```
ALTER TABLE table_name
MODIFY COLUMN column_name
```

Example:

```
ALTER TABLE Employee
ADD DateOfBirth date
```

OUTPUT:

EMP_Id	FirstName	LastName	Designation	City	DateOfBirth
1	Raj	Malhotra	Manager	Mumbai	
2	Henna	Setpal	Executive	Delhi	

DROP COMMAND:

This statement is used to delete a table.

Syntax:

```
DROP TABLE table_name
```

Example:

```
DROP TABLE Employee
```

DATA MANIPULATION LANGUAGE (DML)

DML is set of commands used to,

- Insert data into table
- Delete data from table
- Update data of table.

EMP_Id	FirstName	LastName	Designation	City
1	Raj	Malhotra	Manager	Mumbai
2	Henna	Setpal	Executive	Delhi
3	Aishwarya	Rai	Trainee	Indore

a. INSERT

The INSERT statement is used to insert a new row in a table.

Syntax:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Example:

```
INSERT INTO Employee VALUES (4,'Nihar')
INSERT INTO Employee VALUES (5,'savita')
INSERT INTO Employee VALUES (6,'Diana')
```

OUTPUT:

Emp_Id	FirstName
4	Nihar
5	Savita
6	Diana

b. DELETE

The DELETE statement is used to delete records in a table.

Syntax:

```
DELETE FROM table_name
WHERE some_column=some_value
```

Example:

EMP_Id	FirstName	LastName	Designation	City
1	Raj	Malhotra	Manager	Mumbai
2	Henna	Setpal	Executive	Delhi
3	Aishwarya	Rai	Trainee	Indore

```
DELETE FROM Employee
WHERE LastName='Malhotra' AND FirstName='Raj'
```

OUTPUT:

EMP_Id	FirstName	LastName	Designation	City
2	Henna	Setpal	Executive	Delhi
3	Aishwarya	Rai	Trainee	Indore

c. UPDATE

The UPDATE statement is used to update records in a table.

Syntax:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Example:

EMP_Id	FirstName	LastName	Designation	City
2	Henna	Setpal	Executive	Delhi
3	Aishwarya	Rai	Trainee	Indore
4	Nihar	Sarkar		

```
UPDATE Employee
SET Designation='CEO, City='Mumbai'
WHERE LastName='Sarkar' AND FirstName='Nihar'
```

OUTPUT:

EMP_Id	FirstName	LastName	Designation	City
2	Henna	Setpal	Executive	Delhi
3	Aishwarya	Rai	Trainee	Indore
4	Nihar	Sarkar	CEO	Mumbai

SQL BASIC QUERIES

a. SELECT CLAUSE

This statement is used for various attributes or columns of a table. SELECT can have 2 options as SELECT ALL OR SELECT DISTINCT, where SELECT ALL is default select all rows from table and SELECT DISTINCT searches for distinct rows of outputs.

Syntax:

SELECT * FROM table_name

b. FROM CLAUSE

This clause is used to select a Relation/Table Name in a database.

c. WHERE CLAUSE

This clause is used to put a condition on a query result.

Example:

Ex1: SELECT * FROM Employee

EmpID	FirstName	LastName	Designation	City
1	Raj	Malhotra	Manager	Mumbai
2	Henna	Setpal	Executive	Delhi
3	Aishwarya	Rai	Trainee	Indore

Ex 2: To select only the distinct values from the column named "City" from the table above.

```
SELECT DISTINCT City
FROM Employee
WHERE City='Mumbai'
```

Output:

EmpID	FirstName	LastName	Designation	City
1	Raj	Malhotra	Manager	Mumbai

Aliases

- SQL Aliases are defined for columns and tables.
- Basically aliases are created to make the column selected more readable.

Example:

To select the first name of all the students, the query would be like:

Aliases for columns:

```
SELECT FirstName AS Name FROM Employee;
or
SELECT FirstName Name FROM Employee;
```

In the above query, the column FirstName is given a alias as 'name'.

So when the result is displayed the column name appears as 'Name' instead of 'FirstName'.

Output:

Name
Raj
Henna
Aishwarya
Nihar

Aliases for tables:

```
SELECT e.FirstName FROM Employee e;
```

In the above query, alias 'e' is defined for the table Employee and the column FirstName is selected from the table.

- Aliases is more useful when
- There are more than one tables involved in a query,
- Functions are used in the query,
- The column names are big or not readable,
- More than one columns are combined together

SQL ORDER BY

The ORDER BY clause is used in a SELECT statement to sort results either in ascending or descending order. Oracle sorts query results in ascending order by default.

Syntax

```
SELECT column-list
FROM table_name [WHERE condition]
[ORDER BY column1 [, column2, .. columnN] [DESC]];
```

Database table "Employee";

EmpID	Name	LastName	Designation	Salary	City
1	Raj	Malhotra	Manager	56000	Mumbai
2	Henna	Setpal	Executive	25000	Delhi
3	Aishwarya	Rai	Trainee	20000	Indore

Example:

To select the entire Employee from the table above, however, we want to sort the employee by their last name.

```
SELECT * FROM Employee
ORDER BY LastName
```

Output:

EmpID	Name	LastName	Designation	Sa
1	Raj	Malhotra	Manager	56
2	Henna	Setpal	Executive	25
3	Aishwarya	Rai	Trainee	20

ORDER BY DESC Clause

Using ORDER BY clause of a SELECT statement.

Example:

To select all the Employee from the table above, however, we want to sort the employee descending by their last name.

```
SELECT * FROM Employee
ORDER BY LastName DESC
```

OUTPUT:

EmpID	Name	LastName	Designation	Salary	City
3	Aishwarya	Rai	Trainee	20000	Indore
2	Henna	Setpal	Executive	25000	Delhi
1	Raj	Malhotra	Manager	56000	Mumbai

AGGREGATE FUNCTIONS

SQL aggregate functions return a single value, calculated from values in a column.

Aggregate functions in SQL are as follows:

- **AVG()** – This functions returns the average value
- **COUNT()** - This functions returns the number of rows
- **MAX()** - This functions returns the largest value
- **MIN()** - This functions returns the smallest value
- **SUM()** - This functions returns the sum

Example

StudID	Name	Marks
1	Rahul	90
2	Savita	90
3	Diana	80
4	Heena	99
5	Jyotika	89
6	Rubi	88

AVG() Function

The AVG() function returns the average value of a numeric column.

This function first calculates sum of column and then divide by total number of rows.

Syntax:

```
SELECT AVG(column_name) FROM table_name
```

Example:

Find average Marks of Students from above table.

```
SELECT AVG(Marks) AS AvgMarks FROM Employees
```

The result-set will look like this:

AvgMarks
89.3

COUNT() Function

The COUNT() function returns the number of rows that matches a specified criteria.

Syntax:

```
SELECT COUNT(column_name) FROM table_name
```

Example

```
SELECT COUNT(StudID) AS Count FROM Students
```

Count
6

SUM() Function

The SUM() function returns the total sum of a numeric column.

Syntax

```
SELECT SUM(column_name) FROM table_name
```

Example

Find total of marks scored by students.

Select SUM (Marks) as Sum from Students

OutPut:

SUM
536

MIN() Function

The MIN() function returns the smallest value of the selected column.

Syntax

```
SELECT MIN(column_name) FROM table_name
```

Example

Find minimum scored by students

Select MIN(Marks) as Min from Students

Min
80

MAX() Function

The MAX() function returns the largest value of the selected column.

Syntax

```
SELECT MAX(column_name) FROM table_name
```

Example

Find maximum scored by students

Select MAX(Marks) as Max from Students

Max
90

NESTED SUB-QUERIES

- A query within a query is called Sub-Query.
- Subquery or Inner query or Nested query is a query in a query.
- Sub query in **WHERE Clause (<>, <=>, =, <>)**: It is used to select some rows from main query.
- Sub query in **HAVING Clause (IN/ANY/ALL)**: It is used to select some groups from main query. Subqueries can be used with the following sql statements along with the comparison operators like =, <, >, >=, <= etc.

SYNTAX:

```
SELECT select_Item
FROM table_name
WHERE expr_Operator(SELECT select_item
FROM Table_name)
```

Expression operator can be of 2 types:

1. **Single Row Operator**
2. **Multiple-row Operator**

Single Row Operator

A single-row subquery is one where the subquery returns only one value. In such a subquery you must use a single-row operator such as:

Operator	Description
=	Equal To
<>	Not Equal To
>	Greater Than
>=	Greater Than Equal To
<	Less Than
<=	Less Than Equal To

The single-row operators are used to write single-row subqueries. The table below demonstrates the use of the single-row operators in writing single-row subqueries.

Operator	Query	Example
=	Retrieve the details of employees who get the same salary as the employee whose ID is 101.	SELECT * FROM EMPLOYEES WHERE SALARY=(SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=101);
<>	Retrieve the details of departments that are not located in the same location ID as department 10.	SELECT * FROM DEPARTMENTS WHERE LOCATION_ID <>(SELECT LOCATION_ID FROM DEPARTMENTS WHERE DEPARTMENT_ID=10);
>	Retrieve the details of employees whose salary is greater than the minimum salary.	SELECT * FROM EMPLOYEES WHERE SALARY > (SELECT MIN(SALARY) FROM EMPLOYEES);
>=	Retrieve the details of employees who were hired on or after the same date that employee 201 was hired.	SELECT * FROM EMPLOYEES WHERE HIRE_DATE >= (SELECT HIRE_DATE FROM EMPLOYEES WHERE EMPLOYEE_ID=201);
<	Retrieve the details of employees whose salary is less than the maximum salary of employees in department 20.	SELECT * FROM EMPLOYEES WHERE SALARY < (SELECT MAX(SALARY) FROM EMPLOYEES WHERE DEPARTMENT_ID=20);
<=	Retrieve the details of employees who were hired on or before the same date that employee 201 were hired.	SELECT * FROM EMPLOYEES WHERE HIRE_DATE <=(SELECT HIRE_DATE FROM EMPLOYEES WHERE EMPLOYEE_ID=201);

A multiple row subquery is one where the subquery may return more than one value. In such type of subquery, it is necessary to use a multiple-row operator

The table below describes the multiple-row operators that can be used when writing multiple-row subqueries:

Operator	Meaning
IN	Equal to any value returned by the subquery
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

The multiple-row operators are used to write multiple-row subqueries. The table below demonstrates the use of the multiple-row operators in writing multiple-row subqueries.

Operator	Query	Example
IN	Retrieve department location ID of departments that are located in the same location ID as a location in the UK.	the SELECT DEPARTMENT_ID, DEPARTMENT_NAME, LOCATION_ID FROM DEPARTMENTS WHERE LOCATION_ID IN (SELECT LOCATION_ID FROM LOCATIONS WHERE COUNTRY_ID='UK')
>ALL (Greater than the maximum returned by the subquery)	Retrieve the first name of employees whose salary is greater than the all the salaries of employees belonging to department 20.	SELECT FIRST_NAME FROM EMPLOYEES WHERE SALARY > ALL (SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT_ID=20)
<ALL (Less than the least value returned by the subquery)	Retrieve the first name of employees whose salary is less than all the salaries of employees belonging to department 20.	SELECT FIRST_NAME FROM EMPLOYEES WHERE SALARY < ALL (SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT_ID=20)
>ANY (Greater than the minimum value returned by the subquery)	Retrieve the first name of employees whose salary is greater than the minimum salary of the employees in department 60.	SELECT FIRST_NAME FROM EMPLOYEES WHERE SALARY > ANY (SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT_ID=60)

<ANY	Retrieve the first name	SELECT	FIRST_NAME
(Less than the	of employees whose	FROM	EMPLOYEES
maximum	salary is less than the	WHERE	SALARY < ANY
value returned	maximum salary of	(SELECT	SALARY
by	the employees	in	FROM EMPLOYEES WHERE
subquery)	department 60.		DEPARTMENT_ID=10)

EXISTS CLAUSE

- Exist Clause specifies a sub query to test for the existence of rows.
- Their results type is in BOOLEAN format.
- It Returns TRUE if a sub query contains any rows

Example:

```

SELECT *
FROM suppliers
WHERE EXISTS
(select *
 from orders
 where suppliers.supplier_id = orders.supplier_id);

```

This select statement will return all records from the suppliers table where there is at least one record in the orders table with the same supplier_id.

NOT EXISTS CLAUSE

- The EXISTS condition can also be combined with the NOT operator.

Example:

```

SELECT *
FROM suppliers
WHERE not exists (select * from orders Where
suppliers.supplier_id = orders.supplier_id);

```

This will return all records from the suppliers table where there are **no** records in the *orders* table for the given supplier_id.

NULL VALUES

- NULL values represent missing unknown data.
- By default, a table column can hold NULL values.

- If a column in a table is optional, we can insert a new record or update an existing record without adding a value to this column. This means that the field will be saved with a NULL value.
- NULL values are treated differently from other values.
- NULL is used as a placeholder for unknown or inapplicable values.

"Employee" table:

EmpId	FirstName	LastName	Address	City
1	Hussain	Lakdhwala		Santacruz
2	Elie	Sen	Juhu Road	Santacruz
3	Ranbir	Kapoor		Bhayander

Suppose that the "Address" column in the "Employee" table is optional.

This means that if we insert a record with no value for the "Address" column, the "Address" column will be saved with a NULL value.

IS NULL VALUES

How do we select only the records with NULL values in the "Address" column?

We will have to use the IS NULL operator:

SELECT FirstName, LastName, Address **FROM** Employee
WHERE Address **IS NULL**

Output:

FirstName	LastName	Address
Hussain	Lakdhwala	
Ranbir	Kapoor	

IS NOT NULL VALUES

How do we select only the records with no NULL values in the "Address" column?

We will have to use the **IS NOT NULL** operator:

SELECT LastName,FirstName,Address **FROM** Employee
WHERE Address **IS NOT NULL**

Output:

FirstName	LastName	Address
Elie	Sen	Juhu Road

JOINS

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

Syntax for joining two tables is:

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

If a sql join condition is omitted or if it is invalid the join operation will result in a Cartesian product. The Cartesian product returns a number of rows equal to the product of all rows in all the tables being joined.

Example:

If the first table has 20 rows and the second table has 10 rows, the result will be $20 * 10$, or 200 rows.
This query takes a long time to execute.

Let us use the below two tables to explain the sql join conditions.

Database table "product";

Product_id	Product_name	Supplier_name	Unit_price
100	Camera	Nikon	300
101	Television	LG	100
102	Refrigerator	Videocon	150
103	IPod	Apple	75
104	Mobile	Nokia	50

Database table "order_items";

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

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

- 1) SQL Equi joins
- 2) SQL Non equi joins

1) SQL Equi joins

- It is a simple sql join condition which uses the equal sign as the comparison operator. Two types of equi joins are SQL Outer join and SQL Inner join.

Example:

We can get Information about a customer who purchased a product and the quantity of product.

An equi-join is classified into two categories:

- a) SQL Inner Join
- b) SQL Outer Join

a) SQL Inner Join:

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

Example:

To display the product information for each order the query will be as given below.

Since retrieving the data from two tables, you need to identify the common column between these two tables, which is the product_id.

QUERY:

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


The columns must be referenced by the table name in the join condition, because product_id is a column in both the tables and needs a way to be identified.

b) SQL Outer Join:

- Outer join condition returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables.
- The syntax differs for different RDBMS implementation.
- Few of them represent the join conditions as " **LEFT OUTER JOIN**" and "**RIGHT OUTER JOIN**".

Example

Display all the product data along with order items data, with null values displayed for order items if a product has no order item.

QUERY

```
SELECT p.product_id, p.product_name, o.order_id,
o.total_units
FROM order_items o, product p
WHERE o.product_id (+) = p.product_id;
```

Output:

Product_id	product_name	order_id	total_units
100	Camera		
101	Television	5103	10
102	Refrigerator	5101	5
103	iPod	5102	25

SQL Self Join:

A Self Join is a type of sql join which is used to join a table to it, particularly when the table has a FOREIGN KEY that references its own PRIMARY KEY.

It is necessary to ensure that the join statement defines an alias for both copies of the table to avoid column ambiguity.

Example

```
SELECT a.sales_person_id, a.name, a.manager_id,
b.sales_person_id, b.name
FROM sales_person a, sales_person b
WHERE a.manager_id = b.sales_person_id;
```

2) SQL Non Equi Join:

A Non Equi Join is a SQL Join whose condition is established using all comparison operators except the equal (=) operator.

Like >=, <=, <, >

Example:

Find the names of students who are not studying either Economics, the sql query would be like, (lets use Employee table defined earlier.)

QUERY:

```
SELECT first_name, last_name, subject
FROM Employee
WHERE subject != 'Economics'
```

Output:

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

TRIGGERS

A trigger is an operation that is executed when some kind of event occurs to the database. It can be a data or object change.

Creation of Triggers

- Triggers are created with the CREATE TRIGGER statement.
- This statement specifies that the on which table trigger is defined and on which events trigger will be invoked.
- To drop Trigger one can use DROP TRIGGER statement.

Syntax:

```
CREATE TRIGGER [owner.]trigger_name
ON[owner.] table_name
FOR[INSERT/UPDATE/DELETE] AS
IF UPDATE(column_name)
[{{AND/OR} UPDATE(COLUMN_NAME)...]
{ sql_statements }
```

Triggers Types:

- a. Row level Triggers
- b. Statement Level Triggers

a. Row Level triggers-

A row level trigger is fired each time the table is affected by the triggering statement.

Example:

- If an **UPDATE** statement updates multiple rows of a table, a row trigger is fired once for each row affected by the update statement.
- A row trigger will not run, if a triggering statement affects no rows.
- If **FOR EACH ROW** clause is written that means trigger is row level trigger.

b. Statement Level Triggers

A statement level trigger is fired once on behalf of the triggering statement, regardless of the number of rows in the table that the triggering statement affects, even if no rows are affected.

Example:

- If a **DELETE** statement deletes several rows from a table, a statement level **DELETE** trigger is fired only once.
- Default when **FOR EACH ROW** clause is not written in trigger that means trigger is statement level trigger

Rules of Triggers

- Triggers cannot create or modify Database objects using triggers
 - For example, cannot perform "CREATE TABLE... or ALTER TABLE" sql statements under the triggers
- It cannot perform any administrative tasks
 - For example, cannot perform "BACKUP DATABASE..." task under the triggers
- It cannot pass any kind of parameters
- It cannot directly call triggers
- **WRITETEXT** statements do not allow a trigger

Advantages of Triggers:-

Triggers are useful for auditing data changes or auditing database as well as managing business rules.

Below are some examples:

- Triggers can be used to enforce referential integrity (For example you may not be able to apply foreign keys)
- Can access both new values and old values in the database when going to do any insert, update or delete

Disadvantages of Triggers

- Triggers hide database operations.
- For example when debugging a stored procedure, it's possible to not be aware that a trigger is on a table being checked for data changes
- Executing triggers can affect the performance of a bulk import operation .

Solution for Best Programming Practice

- Do not use triggers unnecessarily, if using triggers use them to resolve a specific situation
- Where possible, replace a trigger operation with a stored procedure or another kind of operation
- Do not write lengthy triggers as they can increase transaction duration; and also reduce the performance of data insert, update and delete operations as the trigger is fired every time the operation occurs.



TRANSACTION MANAGEMENT

Unit Structure

13.0 Objectives

13.1 Introduction

TRANSACTION

- A **transaction** is a logical unit of work that contains one or more SQL statements. A transaction is an atomic unit. The effects of all the SQL statements in a transaction can be either all **committed** (applied to the database) or all **rolled back** (undone from the database).
- A transaction begins with the first executable SQL statement.
- A transaction ends when it is committed or rolled back, either explicitly with a **COMMIT** or **ROLLBACK** statement or implicitly when a DDL statement is issued.
- To illustrate the concept of a transaction, consider a banking database. When a bank customer transfers money from a savings account to a checking account, the transaction can consist of three separate operation:
 - i. Decrement the savings account
 - ii. Increment the checking account
 - iii. Record the transaction in the transaction journal

EXAMPLE:

To illustrate Banking transaction:

Transaction Begins

```
UPDATE savings_accounts
SET balance = balance - 500
WHERE account = 3209;
```

Decrement Savings Account

```
UPDATE checking_accounts
SET balance = balance + 500
WHERE account = 3208;
```

Increment Checking Account

```
INSERT INTO journal VALUES
(journal_seq.NEXTVAL, '1B'
3209, 3208, 500);
```

Record in Transaction Journal

```
COMMIT WORK;
```

End Transaction

Transaction Ends**PROPERTIES OF TRANSACTION:**

Four properties of Transaction: **(ACID PROPERTIES)**

1. Atomicity= all changes are made (commit), or none (rollback).
2. Consistency = transaction won't violate declared system integrity constraints
3. Isolation= results independent of concurrent transactions.
4. Durability= committed changes survive various classes of hardware failure

ATOMICITY

- All-or-nothing, no partial results.
- An event either happens and is committed or fails and is rolled back.
- **EXAMPLE:** In a money transfer, debit one account, credit the other. Either both debiting and crediting.
- If a transaction ends, we say its **commits**, otherwise it **aborts**

3

- Transactions can be incomplete for three reasons:
 1. It can be **aborted** by the DBMS,
 2. A system crash.
 3. The transaction aborts itself.
- When a transaction does not commit, its partial effects should be undone
- Users can then forget about dealing with incomplete transactions
- But if it is committed it should be durable
- The DBMS uses a **log** to ensure that incomplete transactions can be undone, if necessary.

CONSISTENCY

- If the database is in a consistent state before the execution of the transaction, the database remains consistent after the execution of the transaction.

Example:

Transaction T1 transfers \$100 from Account A to Account B. Both Account A and Account B contains \$500 each before the transaction.

Transaction T1

Read (A)
A=A-100
Write (A)
Read (B)
B=B+10

Consistency Constraint

Before Transaction execution $Sum = A + B$
 $Sum = 500 + 500$
 $Sum = 1000$

After Transaction execution $Sum = A + B$
 $Sum = 400 + 600$
 $Sum = 1000$

Before the execution of transaction and after the execution of transaction SUM must be equal.

ISOLATION

- **Isolation** requires that multiple transactions occurring at the same time not impact each other's execution.
- **Example**, if Joe issues a transaction against a database at the same time that Mary issues a different transaction; both transactions should operate on the database in an isolated manner.
- The database should either perform Joe's entire transaction before executing Mary's or vice-versa.
- This prevents Joe's transaction from reading intermediate data produced as a side effect of part of Mary's transaction that will not eventually be committed to the database.
- Note that the isolation property does not ensure which transaction will execute first, merely that they will not interfere with each other.

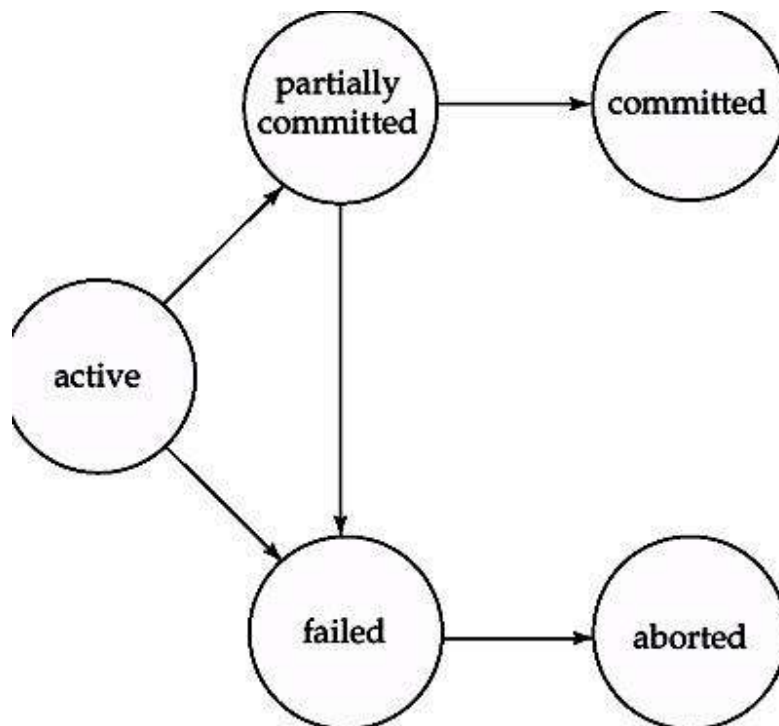
DURABILITY

- **Durability** ensures that any transaction committed to the database will not be lost.
- Durability is ensured through the use of database backups and transaction logs that facilitate the restoration of committed transactions in spite of any subsequent software or hardware failures.

TRANSACTION STATE DIAGRAM

The following are the different states in transaction processing in a Database System.

1. Active
2. Partially Committed
3. Failed
4. Aborted
5. Committed



1. Active

This is the initial state. The transaction stay in this state while it is executing.

2. Partially Committed

This is the state after the final statement of the transaction is executed.

3. Failed

After the discovery that normal execution can no longer proceed.

4. Aborted

The state after the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.

5. Committed

The state after successful completion of the transaction. We cannot abort or rollback a committed transaction.

TRANSACTION SCHEDULE

When multiple transactions are executing concurrently, then the order of execution of operations from the various transactions is known as schedule.

Serial Schedule

Non-Serial Schedule

Serial Schedule

Transactions are executed one by one without any interleaved operations from other transactions.

Non-Serial Schedule

A schedule where the operations from a set of concurrent transactions are interleaved.

SERIALIZABILITY**What is Serializability?**

A given non serial schedule of n transactions is serializable if it is equivalent to some serial schedule.

i.e. this non serial schedule produce the same result as of the serial schedule. Then the given non serial schedule is said to be serializable.

A schedule that is not serializable is called a non-serializable.

Non-Serial Schedule Classification

Serializable

Not Serializable

Recoverable

Non Recoverable

Serializable Schedule Classification

Conflict Serializable

View Serializable

Conflict Serializable Schedule

If a schedule S can be transformed into a schedule S' by a series of swaps of non conflicting instruction then we say that S and S' are conflict equivalent.

A schedule S is called conflict serializable if it is conflict equivalent to a serial schedule.

View Serializable Schedule

All conflict serializable schedule are view serializable.

But there are view serializable schedule that are not conflict serializable.

A schedule S is a view serializable if it is view equivalent to a serial schedule.

Recoverable Schedule Classification

Cascade

Cascadeless

To recover from the failure of a transaction T_i , we may have to rollback several transactions.

This phenomenon in which a single transaction failure leads to a series of transaction roll back is called cascading roll back.

Avoid cascading roll back by not allowing reading uncommitted data.

But this lead to a serial schedule.

