Database Management Systems

Semester V - Computer Engineering
(Savitribai Phule Pune University)

Strictly as per the New Credit System Syllabus (2015 Course)
Savitribai Phule Pune University w.e.f. academic year 2017-2018

Pankaj B. Brahmankar
Director, Phoenix InfoTech, Pune,
Maharashtra, India.

Tech-Max Publications, Pune
Innovation Throughout
Engineering Division

PO265B

Database Management Systems

(Semester V - Computer Engineering, (Savitribai Phule Pune University))

Pankaj B. Brahmankar

First Edition : June 2017

Second Revised Edition : June 2018

This edition is for sale in India, Bangladesh, Bhutan, Maldives, Nepal, Pakistan, Sri Lanka and designated countries in South-East Asia. Sale and purchase of this book outside of these countries is unauthorized by the publisher.

Printed at : Image Offset, Dugane Ind. Area, Survey No. 28/25, Dhayari, Near Pari Company, Pune - 41, Maharashtra State, India. E-mail : rahulshahimage@gmail.com

ISBN 978-93-5224-577-2

Published by

Tech-Max Publications

Head Office : B/5, First floor, Maniratna Complex, Taware Colony, Aranyeshwar Corner, Pune - 411 009. Maharashtra State, India

Ph : 91-20-24225065, 91-20-24217965. Fax 020-24228978.

Email : info@techmaxbooks.com,

Website : www.techmaxbooks.com

[310242] (FID : TP463) (Book Code : PO265B)

Preface

My dear students,

I am extremely happy to come out with this book on **"Database Management Systems"** for the students. This book has been strictly written as per the syllabus. I have divided the syllabus into small chapters so that the topics can be arranged and understood properly. The topics within the chapters have been arranged in a proper sequence to ensure smooth flow of the subject.

I am thankful to Shri. Pradeep Lunawat and Shri. Sachin Shah for the encouragement and support that they have extended. I am also thankful to the staff members of Tech-Max Publications and others for their efforts to make this book as good as it is. We have jointly made every possible efforts to eliminate all the errors in this book. However if you find any, please let me know, because that will help me to improve further.

I am also thankful to my family members and friends for patience and encouragement.

- Pankaj B. Brahmankar

□□□

Syllabus

310242 : Database Management Systems

Teaching Scheme	Credit	Examination Scheme
Theory : 3 Hrs/Week	03	In Sem (Paper) : 30 Marks
		End Sem (Paper) : 70 Marks

Prerequisites Courses

Discrete Mathematics (210241), Data Structures (210243 & 210252)

Companion Course

Database Management System Lab (310247)

Course Objectives

- To understand the fundamental concepts of database management. These concepts include aspects of database design, database languages, and database-system implementation.
- To provide a strong formal foundation in database concepts, technology and practice.
- To give systematic database design approaches covering conceptual design, logical design and an overview of physical design.
- Be familiar with the basic issues of transaction processing and concurrency control.
- To learn and understand various Database Architectures and Applications.
- To learn a powerful, flexible and scalable general purpose database to handle big data.

Course Outcomes

On completion of the course, student will be able to -

- Design E-R Model for given requirements and convert the same into database tables.
- Use database techniques such as SQL & PL/SQL.
- Use modern database techniques such as NoSQL.
- Explain transaction Management in relational database System.
- Describe different database architecture and analyses the use of appropriate architecture in real time environment.
- Use advanced database Programming concepts .

Course Contents

Unit I : Introduction **(07 Hours)**

Introduction to Database Management Systems, Purpose of Database Systems, Database-System Applications, View of Data, Database Languages, Database System Structure, Data Models, Database Design and ER Model : Entity, Attributes, Relationships, Constraints, Keys, Design Process, Entity Relationship Model, ER Diagram, Design Issues, Extended E-R Features, converting E-R & EER diagram into tables. **(Refer Chapter 1)**

Unit II : SQL AND PL/SQL **(07 Hours)**

SQL : Characteristics and advantages, SQL Data Types and Literals, DDL, DML, DCL, TCL, SQL Operators, Tables : Creating, Modifying, Deleting, Views: Creating, Dropping, Updating using Views, Indexes, SQL DML Queries : SELECT Query and clauses, Set Operations, Predicates and Joins, Set membership, Tuple Variables, Set comparison, Ordering of Tuples, Aggregate Functions, Nested Queries, Database Modification using SQL Insert, Update and Delete Queries. **PL/SQL :** concept of Stored Procedures & Functions, Cursors, Triggers, Assertions, roles and privileges , Embedded SQL, Dynamic SQL. **(Refer Chapter 2)**

Unit III : Relational Database Design **(08 Hours)**

Relational Model : Basic concepts, Attributes and Domains, CODD's Rules, Relational Integrity: Domain, Referential Integrities, Enterprise Constraints, Database Design : Features of Good Relational Designs, Normalization, Atomic Domains and First Normal Form, Decomposition using Functional Dependencies, Algorithms for Decomposition, 2NF, 3NF, BCNF, Modeling Temporal Data. **(Refer Chapter 3)**

Unit IV : Database Transactions and Query Processing **(08 Hours)**

Basic concept of a Transaction, Transaction Management, Properties of Transactions, Concept of Schedule, Serial Schedule, Serializability : Conflict and View, Cascaded Aborts, Recoverable and Non-recoverable Schedules, Concurrency Control : Need, Locking Methods, Deadlocks, Time-stamping Methods, Recovery methods : Shadow-Paging and Log-Based Recovery, Checkpoints, Query Processing, Query Optimization, Performance Tuning. **(Refer Chapter 4)**

Unit V : Parallel and Distributed Databases **(07 Hours)**

Introduction to Database Architectures: Multi-user DBMS Architectures, Case study- Oracle Architecture. **Parallel Databases :** Speedup and Scale up, Architectures of Parallel Databases.

Distributed Databases : Architecture of Distributed Databases, Distributed Database Design, Distributed Data Storage, Distributed Transaction : Basics, Failure modes, Commit Protocols, Concurrency Control in Distributed Database. **(Refer Chapter 5)**

Unit VI : NoSQL Database **(08 Hours)**

Introduction to NoSQL Database, Types and examples of NoSQL Database - Key value store, document store, graph, Performance, Structured verses unstructured data, Distributed Database Model, CAP theorem and BASE Properties, Comparative study of SQL and NoSQL, NoSQL Data Models, Case Study-unstructured data from social media. Introduction to Big Data, HADOOP : HDFS, MapReduce. **(Refer Chapter 6)**

310247 : Database Management System Lab

Teaching Scheme	Credits	Examination Scheme
Practical : 4 Hours/Week	02	Practical : 50 Marks
		Term Work : 25 Marks

Companion Course

Database Management System (310242)

Course Objectives

- To develop basic, intermediate and advanced Database programming skills.
- To develop basic Database administration skills
- To percept transaction processing

Course Outcomes

On completion of the course, student will be able to -

- Develop the ability to handle databases of varying complexities
- Use advanced database Programming concepts

Guidelines for Instructor's Manual

The instructor's manual is to be developed as a hands-on resource and reference. The instructor's manual need to include prologue (about University/program/ institute/ department/foreword/ preface etc), University syllabus, conduction & Assessment guidelines, topics under consideration-concept, objectives, outcomes, set of typical applications/assignments/ guidelines, and references.

Guidelines for Student's Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of prologue, Certificate, table of contents, and handwritten write-up of each assignment (Title, Objectives, Problem Statement, Outcomes, software & Hardware requirements, Date of Completion, Assessment grade/marks and assessor's sign, Theory- Concept in brief, Database design, test cases, conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as softcopy.

As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal may be avoided. Use of DVD containing students programs maintained by lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Guidelines for Assessment

Continuous assessment of laboratory work is done based on overall performance and lab assignments performance of student. Each lab assignment assessment will assign grade/marks based on parameters with appropriate weightage. Suggested parameters for overall assessment as well as each lab assignment assessment include- timely completion, performance, innovation, efficient codes, punctuality and neatness.

Guidelines for Practical Examination

Both internal and external examiners should jointly set problem statements. During practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement. The supplementary and relevant questions may be asked at the time of evaluation to test the student's for advanced learning, understanding of the fundamentals, effective and efficient implementation. So encouraging efforts, transparent evaluation and fair approach of the evaluator will not create any uncertainty or doubt in the minds of the students. So adhering to these principles will consummate our team efforts to the promising start of the student's academics

Guidelines for Laboratory Conduction

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The assignment framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. The instructor may set multiple sets of assignments and distribute among batches of students. It is appreciated if the assignments are based on real world problems/applications. Encourage students for appropriate use of Hungarian notation, proper indentation and comments. Use of open source software is to be encouraged. In addition to these, instructor may assign one real life application in the form of a mini-project based on the concepts learned. Instructor may also set one assignment or mini-project that is suitable to respective branch beyond the scope of syllabus.

Set of suggested assignment list is provided in groups- A and B. Each student must perform at least 13 assignments (8-Mandotory plus 4 from remaining 8 assignments) from group A , 5 from group B and 2 mini projects from Group C

Operating System recommended :- 64-bit Open source Linux or its derivative

Programming tools recommended : SQL, PL/SQL, Front End: Java/Perl/PHP/Python/Ruby/.net, Backend : Monod/MYSQL/Oracle, Database Connectivity : ODBC/JDBC

Suggested List of Laboratory Assignments

	Group A : Database Programming Languages - SQL, PL/SQL
1.	Study of Open Source Relational Databases : MySQL.
2.	Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym.
3.	Design at least 10 SQL queries for suitable database application using SQL DML statements : Insert, Select, Update, Delete with operators, functions, and set operator.
4.	Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.
5.	Unnamed PL/SQL code block : Use of Control structure and Exception handling is mandatory. Write a PL/SQL block of code for the following requirements : Schema : 1. Borrower(Rollin, Name, DateofIssue, NameofBook, Status) 2. Fine(Roll_no,Date,Amt) • Accept roll_no & name of book from user. • Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day. • If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. • After submitting the book, status will change from I to R. • If condition of fine is true, then details will be stored into fine table. **Frame the problem statement for writing PL/SQL block inline with above statement.**
6.	Cursors : (All types : Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor) Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped. **Frame the separate problem statement for writing PL/SQL block to implement all types of Cursors inline with above statement. The problem statement should clearly state the requirements.**

7.	PL/SQL Stored Procedure and Stored Function. Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class Write a PL/SQL block for using procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll, Name, Class) **Frame the separate problem statement for writing PL/SQL Stored Procedure and function, inline with above statement. The problem statement should clearly state the requirements.**
8.	Database Trigger (All Types : Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table. **Frame the problem statement for writing Database Triggers of all types, in-line with above statement. The problem statement should clearly state the requirements.**
Group B : Large Scale Databases	
1.	Study of Open Source NOSQL Database : MongoDB (Installation, Basic CRUD operations, Execution)
2.	Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)
3.	Implement aggregation and indexing with suitable example using MongoDB.
4.	Implement Map reduces operation with suitable example using MongoDB.
5.	Design and Implement any 5 query using MongoDB
6.	Create simple objects and array objects using JSON
7.	Encode and Decode JSON Objects using Java/Perl/PHP/Python/Ruby
Group C : Mini Project : Database Project Life Cycle	
1.	Write a program to implement MogoDB database connectivity with PHP/ python/Java Implement Database navigation operations (add, delete, edit etc.) using ODBC/JDBC.
2.	Implement MYSQL/Oracle database connectivity with PHP/ python/Java Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.
3.	Using the database concepts covered in Part-I & Part-II & connectivity concepts covered in Part C, students in group are expected to design and develop database application with following details : **Requirement Gathering and Scope finalization** **Database Analysis and Design** • Design Entity Relationship Model, Relational Model, Database Normalization **Implementation** • Front End : Java/Perl/PHP/Python/Ruby/.net • Backend : MongoDB/MYSQL/Oracle • Database Connectivity : ODBC/JDBC **Testing : Data Validation** Group of students should submit the Project Report which will be consist of documentation related to different phases of Software Development Life Cycle : Title of the Project, Abstract, Introduction, scope, Requirements, Data Modeling features, Data Dictionary, Relational Database Design, Database Normalization, Graphical User Interface, Source Code, Testing document, Conclusion. Instructor should maintain progress report of mini project throughout the semester from project group and assign marks as a part of the term work.

❑❑❑

UNIT I

Chapter 1 : Introduction to DBMS 1-1 to 1-36

Syllabus : Introduction to Database Management Systems, Purpose of Database Systems, Database-System Applications, View of Data, Database Languages, Database System Structure, Data Models, Database Design and ER Model : Entity, Attributes, Relationships, Constraints, Keys, Design Process, Entity Relationship Model, ER Diagram, Design Issues, Extended E-R Features, Converting ER & EER diagram into tables.

UNIT II

Chapter 2 : SQL and PL/SQL 2-1 to 2-46

Syllabus : SQL : Characteristics and advantages, SQL Data Types and Literals, DDL, DML, DCL, TCL, SQL Operators, Tables : Creating, Modifying, Deleting, Views : Creating, Dropping, Updating using Views, Indexes, SQL DML Queries: SELECT Query and clauses, Set Operations, Predicates and Joins, Set membership, Tuple Variables, Set comparison, Ordering of Tuples, Aggregate Functions, Nested Queries, Database Modification using SQL Insert, Update and Delete Queries.

PL/SQL: Concept of Stored Procedures & Functions, Cursors, Triggers, Assertions, Roles and privileges, Embedded SQL, Dynamic SQL.

UNIT III

Chapter 3 : Relational Database Design 3-1 to 3-24

Syllabus : Relational Model : Basic concepts, Attributes and Domains, Codd's Rules. Relational Integrity: Domain, Referential Integrities, Enterprise Constraints. Database Design: Features of Good Relational Designs. Normalization, Atomic Domains and First Normal Form, Decomposition using Functional Dependencies, Algorithms for Decomposition, 2NF, 3NF, BCNF. Modeling Temporal Data.

UNIT IV

Chapter 4 : Database Transactions and Query Processing 4-1 to 4-24

Syllabus : Basic concept of a transaction, Transaction Management, Properties of Transaction, Concept of Schedule, Serial schedule, Serializability : Conflict and View, Cascaded Aborts, Recoverable and Non-recoverable schedules, Concurrency Control : Need, Locking Methods, Deadlocks, Time-Stamping Methods, Recovery Methods : Shadow-paging and Log-Based Recovery, Checkpoints, Query Processing, Query Optimization, Performance Tuning.

UNIT V

Syllabus : Introduction to Database architectures : Multi-user DBMS architectures, Case Study- Oracle Architecture, Parallel Databases : Speedup and Scaleup, Architectures of Parallel Databases. Distributed Databases : Architecture of Distributed Databases, Distributed Database Design, Distributed Data Storage Distributed Transaction : Basics, Failure modes, Commit Protocols, Concurrency Control in Distributed Database.

UNIT VI

Syllabus : Introduction to NoSQL Database, Types and examples of NoSQL Database- Key value store, document store, graph, Performance, Structured verses unstructured data, Distributed Database Model, CAP theorem and BASE Properties, Comparative study of SQL and NoSQL, NoSQL Data Models, Case Study - unstructured data from social media, ntroduction to Big Data, Hadoop : HDFS, MapReduce.

❑❑❑

Suggested List of Laboratory Assignments

❑❑❑

UNIT I

CHAPTER 1

Introduction to DBMS

Syllabus

- Introduction to Database Management Systems
- Purpose of Database Systems
- Database-System Applications
- View of Data
- Database Languages
- Database System Structure
- Data Models
- Database Design and ER Model : Entity, Attributes, Relationships, Constraints, Keys, Design Process, Entity Relationship Model
- ER Diagram
- Design Issues
- Extended E-R Features
- Converting ER & EER diagram into tables

Syllabus Topic : Introduction to Database Management Systems

1.1 Introduction to Database Management Systems

- **Data :** Data is the information which has been translated into a form that is more convenient to process or move.
- **Database :** The collection of related data is termed as Database which is organized in such a way that it can be easily retrieved and managed.
- **Database Management System**
  - A Database Management System (DBMS) is system software which manages the data. It can perform various tasks like creation, retrieval, insertion, modification and deletion of data to manage it in a systematic way as per requirement.
  - Database systems are designed to manage large amount of data by providing security from accidental crash of system and unauthorized access. DBMS provides convenient and efficient environment which used to handle the data.

Syllabus Topic : Purpose of Database Systems

1.2 Purpose of Database Systems

- Programming languages like Java, .Net are used to develop customized software's. Every software or application has its data to be stored permanently.
- Programming languages cannot store data permanently. For this purpose we have to use the Database Management System. The DBMS plays a significant role in storing and managing data.
- In an application we store data in DBMS and for operations like insertion, modification or deletion we write code in programming languages i.e. software is usually created with the help of both Programming language and Database.
- When the application is executed on client side, the client or user interacts with interface of application which is created in programming language.

- The database always remains backside and do not come in front of the user. Hence the database is known as backend while programming language is termed as frontend.
- To understand the purpose or need of database system, we need to study the previous option to store data which is called as File Processing System.

1.2.1 File Processing System

- In our day to day life, number of times we need to store data in such way that it should be easily accessible whenever required.
- The data may be of bank transaction details, daily expenses, employee details, product details etc. Before computers such data was stored with the help of papers.
- After invention of computers, it becomes easy to store data with the help of files. In the early days, database applications were built on top of file systems.
- Traditional File Processing System is a computer based system in which all the information is stored in various computer files.
- It stores data in a systematic way that the different departments of an organization can store their data in set of files which helps to manage and differentiate the data.
- Initially the Traditional File Processing System seems to be useful but as the requirement of data processing and the size of data increases, the drawbacks of this system comes in picture.

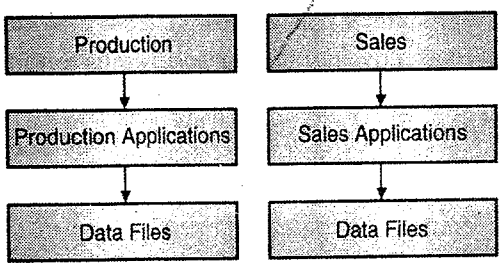


Fig. 1.2.1

1.2.2 Drawbacks of Traditional File Processing Systems

Consider an example of IT company database system where the data of all the employees is stored. In a professional IT firm the tasks are always done as teamwork. Different teams are assigned for different projects. Hence the data is stored in different files so as to differentiate it.

1. Data Redundancy

- Sometimes as per requirement same data may be stored in multiple files. Consider an employee having record in both Employee and Team files. The name and address of employee is stored in both of these files.
- Means the data get duplicated. If such data increases, it leads to higher storage and access cost. This duplication of data in various files is termed as data redundancy.
- In traditional file system, it is very difficult to avoid this data redundancy.

2. Data Inconsistency

- When data is to be updated the data redundancy may lead to data inconsistency. Data inconsistency occurs when data is not updated in all the files simultaneously.
- For example if the designation of employee get changed, then the respective changes should be made in both Employee and Team file. If for some reason, it is not done, then it leads to data inconsistency.
- Because for the sample employee, we may get different information which may create problems in the processing of data.

3. Limited Data Sharing

- It is difficult to share data in traditional file system. Each application has its own private files and users have little choice to share the data outside their own applications.
- To share data, we have to write complex programs.

4. Difficulty in Accessing Data

- The need of data access varies time to time. Means different types of information is needed at different situations.
- For example just consider that we want to retrieve the data of employees who do not have taken any leave throughout the quarter. In such case we have two options.
- We can access the data by manual method or we have to write an application program to

retrieve such customized data. Both the options are not convenient as both of them leads to wastage of time.

- If we do it, then also it may be possible that after some time we may require data with some another filter criteria. The data retrieval for customized information becomes difficult because the conventional files system does not provide any efficient and convenient way to retrieve the data.

5. Data Dependence

- In the files, data is stored in some specific format tab, semicolon or comma. If the format of any of the file is changed, then we have to make changes in program which processes the file.
- But sometimes there may be many programs related with the same file. In such case changes in all such programs should be done. Missing changes in single program may lead to failure of whole application.

6. Poor Data Control

- The Traditional File System does not have centralized data control; the data is de-centralized or distributed. In this system the same field may have different names in files of different departments of an organization.
- This situation may lead to different meaning of same data field in different context or same meaning for different fields. This causes poor data control.

7. Problem of Security

- It is very difficult to enforce security checks and access rights in a traditional file system. To the file we can set password protection.
- But what if we have to give access to only few records in the file? For example, in our database system, the project manager should be able to see all the data regarding teams under him.
- The team leader should be able to see data about his specific team. But payment details of one project manager should not be accessible to his team members or any another project manager. In the conventional file processing system, the application programs are added in ad hoc manner (for specific purpose) which makes it difficult to enforce security constraints.

8. Concurrency Problems

- Concurrency means access of same data by multiple users at the same time. This is very important aspect as it leads to increase in performance of a system and faster response. Many advanced systems allow the concurrent access and manipulation of data.
- For example, in our system, consider a record of an employee is accessed and updated by multiple users simultaneously at a time. This may lead to inconsistency of data, if the concurrency is not controlled in a proper manner.
- In another example if multiple transactions are make updations on a same bank account, then it may show incorrect balance, if any other transactions try to access balance amount in between.
- It is very difficult to implement concurrency control mechanism on file processing system, which leads to incorrect or wrong data retrieval.

9. Poor Data Modelling of Real World

- It is difficult for File Processing System to represent the complex data and interfile relationships. This results in poor data modelling properties.
- That means the real world applications are difficult to implement using File Processing System.

10. Data Isolation

- It is difficult to store the entire data in a single file. It is distributed in different files as per the category.
- These files may be in different formats because of which it becomes difficult to write application programs to access the desired data from these files.

11. Integrity Problems

- Every enterprise has its own constraints while maintain data in the files. Suppose in employee files the employee ID must start with 'E'. Such constraints can be added while writing application programs.

- But later on if any new constraints are introduced by the enterprise, and then it becomes difficult to add these constraints again. The File processing system does not provide any functionality to handle this situation.

12. Atomicity Problem

- Failure in a computer system may occur any time. When failure occurs, if any transaction is in its midway then it may lead to some incorrect data updation in the system.
- Consider another example of bank transaction where some amount is transferred from account A to account B. Initially the balance from account A is accessed and debited by Rs. 1000. Then we are going to credit it in account B. But before that system crash occurs which halts the transaction.
- Now this situation leads to incorrect data updation in the balance of account A. In file processing system, it very difficult to handle such situation to maintain the atomicity of database. The purpose of Database Management System is to solve all these problems and give functionality to store and manage data in efficient and convenient way.

1.2.3 Advantages of Database Management System

1. Controlling Data Redundancy

- In File Processing System the different applications has separate files for data storage. In this case, the duplicated copies of the same data are created at many places.
- In DBMS, all the data of an organization is integrated into a single database.
- The data is recorded at only one place in the database and it is not duplicated. For example, the Employee file and the Team file contain several items that are identical.
- When they are converted into database, the data is integrated into a single database so that multiple copies of the same data are reduced to-single copy.
- Controlling the data redundancy helps to save storage space. Similarly, it is useful for retrieving data from database using queries.

2. Data Consistency

- The data consistency is obtained by controlling the data redundancy, If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users.
- For example if there is change in designation of employee, then the changes are made in single centralized file which is available to all the users.

3. Sharing of Data

- In DBMS, data can be easily shared by different applications. The database administrator manages the data and gives rights to users to access the data.
- Multiple users can be authorized to access the same data simultaneously. The remote users can also share same data.

4. Data Independence

- In DBMS we can completely separate the data structure of database and programs or applications which are used to access the data.
- This is called as data independence. If any changes are made in structure of database then there is no need to make changes in the programs. For example you can modify the size or data type of a data items (fields of a database table) without making any change in application program.

5. Data Control

- The DBMS provides centralized data storage. Hence keeping control on data is very much easy as compared to Traditional File Processing System.
- As data is common for all the application, no possibility of any confusion or complication.

6. Security

- In DBMS the different users can have different levels of access to data based on their roles. In the college database, students will have access to their own data only, while their teachers will have access to data of all the students whom they are teaching.

- Class teacher will be able to see the reports of all the students in that class, but not other classes. The principal will have access to entire data.
- Similarly, in a banking system, individual operator and clerk will have limited access to the data while the bank manager can access the entire data.
- All these levels of security and access are not allowed in file system.

7. Control over Concurrency

- In a computer file-based system, if multiple users are accessing data simultaneously, it is possible that it may lead to some irrelevant data generation. For example, if both users attempt to perform update operation on the same record, then one may overwrite the values recorded by the other.
- Most database management systems have sub-systems to control the concurrency so that transactions are always recorded with accuracy.

8. Data Modelling of Real World

- The DBMS has many functionalities are provided to represent the complex data and interfile relationships.
- This helps to map the database with real world applications.

1.2.4 Disadvantages of Database Management System

1. Increased Costs

- To install Database Systems, we require standard software and hardware. Also to handle the Database System, highly skilled personnel are required.
- The cost of maintaining the software, hardware, and personnel required to operate and manage the database system is more.
- The cost of training, license, and regulation compliance also increases the overall expenses.

2. Complexity

- Sometimes because of higher functionality expectations, the design of Database may become very complex.
- To utilize such database with complete efficiency, all the stakeholders like database designers, developers, database administrators and end-users must understand the functionality.
- Failure in understanding the system can lead to wrong design decisions, because of which serious consequences for an organization may occur.

3. Size

The DBMS becomes extremely large piece of software because of the complexity of functionality occupying large amount of disk space and requiring substantial amounts of memory to run efficiently.

4. Frequent Upgrade/Replacement Cycles

- New functionalities are often added into DBMS by their vendors. These new features often come bundled in new upgraded versions of the same software.
- Sometimes these versions require hardware upgrades which increases expenses. Also work to train database users and administrators to properly use and manage the new features get increased.

5. Higher Impact of a Failure

- The DBMS is placed at centralized location which increases the vulnerability of the system.
- That means the DBMS may get attacked and harmed. Since all users and applications rely on the centralized database, the failure of any component can bring operations to a halt.

6. Performance

- Usually, a File Based system is written for a specific application. Hence, the performance is generally very good.
- While the DBMS is written to be more general, to cater for multiple applications rather than any specific one.
- Because of which some applications may not run as fast as they used to.
- There are number of advantages of DBMS over File System.

1.2.5 Difference between File Processing and DBMS

SPPU Dec. 13, May 14, Dec. 15

University Questions

Q. Explain significant difference between File Processing and DBMS. **(Dec. 2013, 6 Marks)**

Q. Explain the advantages of DBMS over normal file system in detail. **(May 2014, 8 Marks)**

Q. List significant difference between File Processing and DBMS. **(Dec. 2015, 5 Marks)**

Sr. No.	File Processing System	Database Management System
1.	Duplicate data may exist in multiple files which lead to data redundancy.	The data is integrated into a single database which avoids data redundancy.
2.	Data inconsistency occurs when data is not updated in all the files simultaneously.	The data consistency is obtained by controlling the data redundancy
3.	It is difficult to share data in traditional file system.	In DBMS, data can be easily shared by different applications.
4.	In the files, data is stored in specific format. If the format of any of the file is changed, then we have to make changes in program which processes the file.	In DBMS we can completely separate the data structure of database and programs or applications which are used to access the data.
5.	The Traditional File System does not have centralized data control; the data is de-centralized or distributed.	The DBMS provides centralized data storage. Hence keeping control on data is very much easy.
6.	It is very difficult to enforce security checks and access rights in a traditional file system.	In DBMS the different used can have different levels of access to data based on their roles which provides strong security to data.
7.	Concurrency problems means updation of same data by multiple users may generate irrelevant results.	DBMS have sub-systems to control the concurrency.
8.	It is difficult for File Processing System to represent the complex data and interfile relationships. This results in poor data modelling.	The DBMS has many functionalities are provided to represent the complex data and interfile relationships. This helps to map the database with real world applications.

Syllabus Topic : Database-System Applications

1.3 Database-System Applications

For any enterprise, its data is very important which helps to manage the business as well as decide some strategies to survive and grow the business in this competitive world. A Database Management System is a computerized record-keeping system. It works as a container for collection of computerized data files.

The overall purpose of DBMS is to provide functionality to the users to create, store, retrieve and update the information contained in the database as per requirement. Information can be of an individual or an organization.

Databases touch all aspects of our lives. Some of the major **areas of application** are as follows :

- **Telecom :** In Telecom sector database is maintained to keep track of the information about calls made, customer details, network usage etc. Without using database systems it is difficult to maintain such huge amount of data which keeps track of every second.
- **Banking :** In banking sector the data related to transactions of customers is very huge. Such data can be comfortably stored in the Database Management System. The DBMS is also used for storing customer personal information, tracking day to day credit and debit transactions, generating bank statements etc.

- **Industry :** In industry database management plays an important role. In departments like production, sales or account it is very essential to store the data in systematic format. For example in production department the data about manufactured products is stored which is very useful for sales department to keep track of orders.
- **E-commerce :** There are number of online shopping websites such as ebay, Flipkart Amazon, etc. These sites store the information about various products, user addresses, their preferences and credit details. It can be implemented using Database Management System only.
- **Airlines :** In airlines, the data of ticket booking, flight schedule, employee details, and customer detaiis is very huge. Such data can be managed using the Database Management System.
- **Education System :** In education sector, the schools, colleges, private institutes have to store data of students, teachers, exam schedules, accounts etc which can be handles by Database Management System.
- **Railway Reservation System :** Database Management System helps in keeping record of ticket booking, departure and arrival status of train etc.
- **Library Management System :** In the library there are thousands of books which make it very difficult to keep record of all the books in a register. The DBMS can be used to maintain this information about book issue dates, names of the books, authors and availability of the book.
- **Social Media Sites :** The social media websites are used to share our views and connect with our friends. Daily millions of users signed up for these social media accounts like Facebook, Whatsapp, Twitter, and Google plus. The data of these millions of users and their chats can store using Database Management System.

Syllabus Topic : View Of Data

1.4 View of Data

As we have seen a data base system is collection of related data and system software which manages the data. The data is generally stored in a detailed and complex manner. It is important to provide an abstract view of data to the user.

To understand the view of data, first we have to learn the concept of abstraction.

1.4.1 Abstraction

- Abstraction is an important feature of Database Management System. Extracting the important data by ignoring the remaining irrelevant details is known as **abstraction**.
- Database systems are usually made-up of complex data structures as per their requirements.
- To make the user interaction easy with database, the internal irrelevant details can be hidden from users. This process of hiding irrelevant details from user is called data abstraction.
- The complexity of database can be hiding from user by different level of abstraction as follows.

1.4.1.1 Levels of Abstraction SPPU - Dec. 13

University Question

Q. Explain in detail the different levels of abstraction. **(Dec. 2013, 4 Marks)**

There are three levels of abstraction :

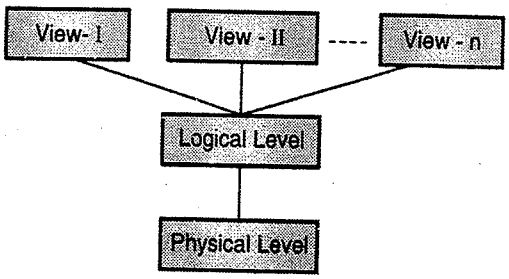


Fig. 1.4.1

1. Physical level

- In data abstraction Physical level is the lowest level. This level describes how the data is actually stored in the physical memory.
- The physical memory may be hard disks, magnetic tapes, etc. In Physical level the methods like hashing are used for organization purpose.
- Developer would know the requirement, size and accessing frequency of the records clearly in this level which makes easy to design this level.

2. Logical level

- This is the next higher level of abstraction which is used to describe what data the database stores,

and what relationships exist in between the data items. The logical level thus describes an entire database in terms of a small number of relatively simple structures.

- Although implementation of the simple structures at the logical level may involve complex physical level structures, the user of the logical level does not need to be aware of this complexity. This is considered as physical data independence.
- Database administrators use the logical level of abstraction to decide what information to keep in a database.

3. View level

SPPU - May 13

University Question

Q. Explain the need of view. **(May 2013, 3 Marks)**

- It is the highest level of data abstraction. This level describes the user interaction with database system. In the logical level, simple structures are used but still complexity remains because in the large database various type of information is stored.
- Many users are not aware of technical details of the system, and also they need not to access whole information from the database. Hence it is necessary to provide a simple and short interface for such users as per their requirements. Multiple views can be created for same database for multiple users.

  Example : Consider that we are storing information of all the employees of an organization in employee table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are usually hidden from the developer.
- The records can be described as fields and attributes along with their data types at the **logical level**. The relationship between these fields can be implemented logically. Usually the developer works at this level because they have knowledge of such things about database systems.
- End user interacts with system with the help of GUI and enters the details at the screen at **view level**. User is not aware of how the data is stored and what data is stored; such details are hidden from them.

1.4.2 Schema

- The design of a database is called the schema. To understand schema we can consider an example of a program of an application. A variable or array declared with its structure (data type and/or size) is schema. The changes in schema or not frequent.
- **Types of Schema** : According to the level of abstraction, the database schema is divided into three types : Physical schema, Logical schema and View schema.

  (i) Physical schema is the design of a database at physical level, i.e. how the data stored in the blocks of storage is described in this level.

  (ii) Logical schema is the design of database at logical level. Developers and database administrators work at this level. Here the data can be described as certain types of data records gets stored in data structures, however the internal details like the implementation of data structure are hidden at this level.

  (iii) View schema refers to design of database at view level. This usually describes the end user interaction with database systems. There may be multiple schemas at view level.

1.4.3 Instance

In database changes are quite frequent i.e. insertion, deletion or updation are the frequent operations on database. The data is stored in the database at particular moment is called as instance of the database. In the example of application program, the value of a variable at particular time or situation is called as instance of database schema.

Syllabus Topic : Database Languages

1.5 Database Languages

In general in a database system, the Data Definition Language is used to specify schemas (design) of a database while the data manipulation language is used to fire queries (commands) on database in order to manipulate it. Both are the main pillars of database language like SQL.

With DDL and DML the database system also has the languages like DCL and TCL for different functionalities. The database languages are categorized as DDL, DML, DCL and TCL.

1.5.1 Data Definition Language (DDL)

- The DDL specify the schema of database by set of definitions.
- This language allows the users to define data and their relationship to other types of data. It is used to create data tables, dictionaries, and files within databases.
- The DDL is also used to specify the structure of each table, set of associated values with each attribute, integrity constraints, security and authorization information for all the tables and physical storage structure of all the tables on the disk.
- The data values stored in the database should satisfy some constraints for consistency purpose. For example the salary of an employee should never be negative or the employee id should start with 'E' etc. The Data Definition Languages provides functionality to specify such constraints.

1.5.2 Data Manipulation Language (DML)

- The Data Manipulation Language (DML) is used for accessing and manipulating data in a database. DML provides a set of functionalities to support the basic data manipulation operations on the data stored in the database.
- The DMS is basically of two types.
  1) **Procedural DML** – There is a requirement of user to specify which data is required and how get this data.
  2) **Declarative DML** – Here is also requirement of user to specify which data is required and without specifying how get this data. This is easier to understand and useful than procedural DML.
- A concept of query is used for processing. Query is a statement or command which requests the retrieval of information from the database. The part of DML which involved information retrieval is known as query language. The three levels of abstraction are used in defining structure of data as well as to manipulate the data.

Syllabus Topic : Database System Structure

1.6 Database System Structure

SPPU Dec. 13

University Question

Q. Explain component and overall structure of DBMS. **(Dec. 2013, 10 Marks)**

- DBMS (Database Management System) acts as an interface between the user and the database.
- The user requests the DBMS to perform various operations (retrieve, insert, delete and update) 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 as shown below :

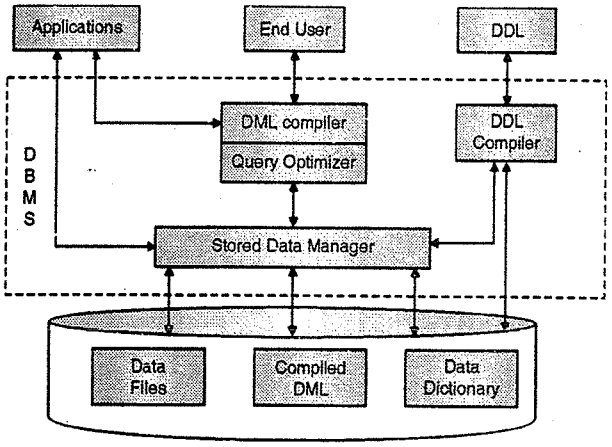


Fig. 1.6.1 : Database System Structure

Structure of DBMS

The Structure of DBMS contains following components :

1. DDL Compiler

- The DDL Compiler converts DDL commands into set of tables containing metadata stored in a data dictionary.
- The metadata information is 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 retrieve, insert, update, delete etc. from the application program are sent to the DML compiler for compilation. It converts these commands into object code for understanding of database.
   - 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 converts the requests received from query optimizer to machine understandable form. It makes actual request inside the database.
     - Controls DBMS information access that is stored on disk.
     - It controls handling buffers in main memory.
     - It enforces constraints to maintain consistency and integrity of the data.
     - It 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 - 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 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 for the use of the system.
   - It helps in documenting 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.
5. **Data File**

   It contains the data portion of the database i.e. it has the real data stored in it. It can be stored as magnetic disks, magnetic tapes or optical disks.
6. **Compiled DML**
   - The DML complier converts the high level Queries into low level file access commands known as compiled DML.
   - Some of the processed DML statements (insert, update, delete) are stored in it so that if there is similar requests, the data can be reused.
7. **End Users**

   They are the real users of the database. They can be developers, designers, administrator or the actual users of the database.

Syllabus Topic : Data Models

1.7 Data Models

Basic Concept of Data Models

The process of analysis of data object and their

relationships to other data objects is known as data modeling. It is the conceptual representation of data in database. It is the first step in database designing. Data models define how data is connected to each other and how they are processed and stored inside the system. A data model provides a way to describe the design of a database at the physical, logical and view levels.

1.7.1 Types of Data Models SPPU - May 13

University Question

Q. Explain various Data Models used in DBMS. **(May 2013, 10 Marks)**

There are different types of data models :

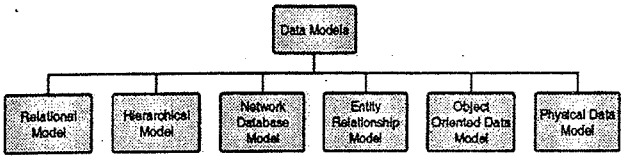


Fig. 1.7.1

1.7.1.1 Relational Model

- The relational model is developed by E.F. Codd. Relational database is a type of record-based relations.
- Relational database is an attempt to simplify the data structure by making use of tables. Tables are used to represent the data and their relationships. Table is a collection of rows and columns. Tables are also known as relations.
- Records are known as tuples and fields are known as attributes.
- The relational model is called as record based model because the database is structured in fixed format records of different types. A record is consists of fields or attributes.
- In the relational model, every record must have a unique identification or key based on the data.
- In following table Stud_ID is the key through which we can identify the record uniquely in the relation. Relational data model is the most widely used record-based data model.

Key

Tuple →

Stud_ID	Stud_Name	DOB
101	Prajakta	03/03/1995
102	Rakesh	13/01/1996
103	Rahul	16/08/1995

Advantages of Relational Data Model

a) **Supports SQL**
   - For accessing the data in Relational data model we have a language known as Structured Query Language (SQL).
   - This language is used to access, insert, update or delete the data from the table. By using relational data model we can execute the complex queries.

b) **Flexible**
   - We can easily manipulate the information which is linked with various tables.
   - We can extract the information from different table simultaneously by using this model.

1.7.1.2 Hierarchical Model

- A data model in which the data is organized into a tree structure is known as hierarchical data model.
- Hierarchical data model structure contains parent-child relationship where root of the tree is a parent which then branches into its children. It is another type of record based data model.
- The data is stored in the form of records. These records are connected to one another.
- A record is a collection of fields; each field contains only one value. The hierarchical data models represent data according to its hierarchy.

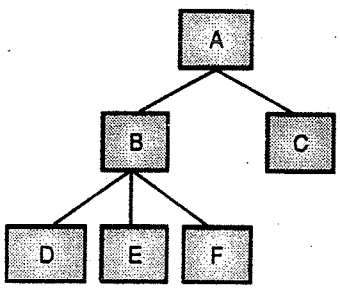


Fig. 1.7.2 : Hierarchical Model

- In hierarchical model there is a business rule. The rule is that, one parent node can have many child nodes but one child nodes can have only one parent node. This type of model is not in use currently.

Advantages of Hierarchical Model

a) **Simple to understand :** Due to its hierarchical structure it is easy to understand. Most of the time data have hierarchical relationship. Therefore, it is easy to arrange the data in that manner.

b) **Database Integrity :** In hierarchical data model there is always a parent child association between different records on different level. Due to this inherent structure integrity gets maintained.

c) **Efficient :** The performance of this model is very efficient due to its hierarchical structure when database contain large amount of data which has various related records.

1.7.1.3 Network Database Model

- It is extended type of hierarchical data model. This data model is also represented as hierarchical, but any child in the tree can have multiple parents.
- In network data model there is no need of parent child association. There is no downward tree structure.

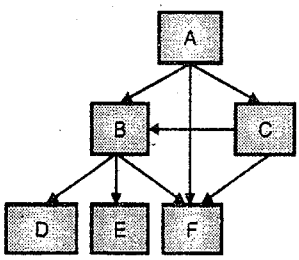


Fig. 1.7.3 : Network Model

- It is the flexible way of representing the objects and their relationship. A network data model allows multiple records linked in the same file.
- Basically, network database model forms a network like structure between the entities.

Advantages of Network Model

a) **Design is simple :** The network model is simple and easy to design and understand. There is no complex structure in network model.

b) **It has capability to handle various relationships :** The network model can handle the one to many and many to many relationships which is useful to develop the database.

c) **Easy to access :** The data access is easy and flexible than the hierarchical data model. In network model there is no any hierarchy in the objects and their relations therefore it is very easy to access the data in network model.

1.7.1.4 Entity Relationship (E-R) Model

- This model describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types. In the next Sections 1.10 and 1.11 we will study this model in detail.

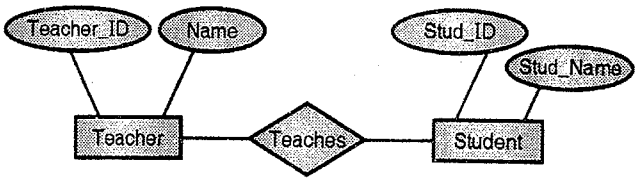


Fig. 1.7.4 : ER Model

Advantages of Entity Relationship Model

a) **Simple Design :** The ER model is simple and easy to design. It shows the logical view of the data and its relationships. This model is easy to understand.

b) **Effective representation :** The presentation of Entity Relationship Model is very simple and effective. The programmer and designer can easily understand the flow of the system by referring the ER Model.

c) **Connected with Relational Model :** The Entity Relationship Model is connected with the relational model. Due to this advantage we can develop a well-structured design.

1.7.1.5 Object Oriented Database Model

- Object oriented data model is nothing but the collection of objects or elements with its methods.
- Now a days all advanced programming languages supports the concepts of Object Oriented Programming.
- This model is based on the Object-oriented paradigm. This paradigm is defined for the database.
- It is extension to E-R model with integration of concepts like object, method and encapsulation.
- Applications of data modelling contain the key concepts of object-oriented.

Advantages of Object- Oriented Data Model

a) Object oriented data Model supports inheritance. Due to this we can reuse attributes and functionalities. It reduces the cost of maintaining data multiple times.

b) Due to inheritance, and encapsulation this model become more flexible.

c) We can bind each class with its attributes and functionality. It helps in representing the real world objects.

d) We can see each object as a real world entity in this model. Hence it is more understandable.

1.7.1.6 Physical Data Model

- The description of data storage in the computer in the form of information is provided by Physical data models.
- Physical data model shows the table structure which includes column name, column data, constraints and relationship between tables. Physical data models define all the components of logical database.

Advantages of Physical Model

a) We can specify the all tables and columns using physical data model.

b) We can use foreign key to join the tables. By using foreign key we can identify the related data from the different tables.

c) By using physical data model we can convert entity into the table.

Syllabus Topic : Database Design and ER Model

1.8 Database Design and ER Model

1.8.1 Database Design

In the creation of database system, the design of database is the most important and initial part. The overall success of the system is completely depends upon the design of database. The design is mainly focused on the security and access of data by the application program. The requirements of user play an important role in database design.

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

- Determine the data to be stored in the database.
- Determine the relationships between the different data elements.
- Superimpose a logical structure upon the data on the basis of these relationships.

Determine the data to be stored in the database

- Number of times, the database designer is a person with expertise in the area of database design. It is not necessary that he should be expertise in the domain from which is the data is retrieved e.g. financial information, biological information etc.
- Hence it is important that the data to be stored in the database should be determined in cooperation with the domain expertise that has knowledge that which data must be stored within the system.
- This process is a part of requirements analysis and required that the database designer should get the required data from domain expertise.
- The reason is that number of times the domain expertise cannot express exactly what their system requirements for the database are unfamiliar to thinking in terms of the discrete data elements which are needed to be stored.
- Data to be stored can be determined by Requirement Specification. In the life cycle of software it is known as requirement gathering phase.

Determining data relationships

- Once a database designer is aware of the data which is to be stored within the database, they must then determine where dependency is within the data. Sometimes when data is changed you can be changing other data that is not visible.
- For example, in a list of names and addresses, assuming a situation where multiple people can have the same address, but one person cannot have more than one address; the address is dependent upon the name.
- When provided a name and the list the address can be uniquely determined; however, the inverse does not hold - when given an address and the list, a name cannot be uniquely determined because multiple people can reside at an address.
- Because an address is determined by a name, an address is considered dependent on a name.

Logically structuring data

- Once the relationships and dependencies amongst the various pieces of information have been determined, it is possible to arrange the data into a logical structure which can then be mapped into

the storage objects supported by the database management system.

- Now the important part in design is that how to represent things like person, product, place etc. These things can be represented in term of entities. The various entities are related with each other by one or other way.

In database design schema, it is necessary to avoid two major problems.

1. **Redundancy**
   - Sometimes as per requirement same data may be stored in multiple files. Consider an employee having record in both Employee and Team files.
   - The name and address of employee is stored in both of these tables. Means the data get duplicated. If such data increases, it leads to higher storage and access cost.
   - This duplication of data in various files is termed as data redundancy. Such redundancy can also occur in relational schema also. In the schema of the database, such information may be repeatedly shown.
2. **Incompleteness**
   - An incomplete design is very difficult to model. Suppose in a IT enterprise database system, if a department like R& D is there, to which no employee is still appointed, then it becomes very difficult to represent information about this department.

1.8.2 E-R Model

- The Entity Relationship (E-R) model allows specifications of an enterprise schema and represents the overall logical structure of the database.
- Now a days the database related to real world applications becomes very vast and complex. Representing relations between the different elements of the database becomes difficult.
- ER Model simplifies this task. It is nothing but the design technique for database. It is a graphical technique which helps to understand and organize the complex data which should not depend upon the actual database implementation.
- The real world objects can be easily mapped with entities of E-R model.
- In Entity Relationship Model a graphical representation of a database system is generated. Diagrams are used in this model. These diagrams are known as entity-relationship diagrams, ER diagrams or ERDs.
- Basic concepts of ER Model are as follows :

Entity, attribute, Relationship, constraints and keys.

Syllabus Topic : Entity

1.8.2.1 Entity and Entity Set

- The E-R model consists of entities and relationships between those entities. An entity is nothing but a thing having its own properties. These properties helps to differentiate the object (entity) from other objects.
- An entity is a thing that exists either physically or logically. An entity may be a physical object such as a house or a car, or an entity may be a logical concept like an event such as a house sale or a car service, or a concept such as a customer transaction or order.
- An entity set is a set of entities which share the same properties. In a Company employee is the entity set which has similar properties like Employee_ID, emp_name, salary etc.
- There is difference between an entity and an entity-type. An entity-type is a category. An entity, strictly speaking, is an instance of a given entity-type. There are usually many instances of an entity-type
- There are two types of entities in Database management system.
  1. **Strong Entity or Regular Entity**
     - If an entity having it's own key attribute specified then it is a strong entity. Key attribute is used to indentify that entity uniquely among set of entities in entity-set.
     - **Example :** In a parent/child relationship, a parent is considered as a strong entity.
     - Strong entity is denoted by a single rectangle.

- The relation between two strong entities is denoted by a single diamond simply called relationship.

2. Weak Entity

- The entity which does not have any key attribute is known as weak entity. The weak entity has a partial discriminator key. Weak entity depends on the strong entity for its existence. Weak entity is denoted with the double rectangle.
- **Example :** In a parent/child relationship, a child is considered as a weak entity which is completely depends upon the strong entity 'parent'.

Comparison of an Object is OOP and Entity in E-R Model

SPPU - May 13

University Question

Q. How does the concept of an object in the object oriented model differ from the concept of an entity in the E-R model ? **(May 2013, 8 Marks)**

Sr. No.	Entity	Object
1.	An entity is a thing that exists either physically or logically. An entity may be a physical object such as a house or a car (they exist physically), an event such as a house sale or a car service, or a concept such as a customer transaction or order (they exist logically as a concept)	Object in Object oriented programming is nothing but anything which has its own properties. E.g. flower is an object having properties like color, fragrance etc.
2.	Even if data in two entity instances is the same, we don't deem them as equivalent.	Objects don't have an identifier field and if two objects have the same set of attributes we can treat them interchangeably.
3.	Entities live in continuum that means they have a history of what happened to them and how they changed during their lifetime.	Objects, at the same time, have a zero lifespan. We create and destroy them with ease.
4.	Entities are mutable. That means we can change them.	Objects should be immutable that means we cannot change them rather we construct a new object based on the existing object.

Syllabus Topic : Attributes

1.8.2.2 Attribute

- An attribute is a characteristic of an entity. Entities are represented by means of their attributes. All attributes have their own specific values. For example, an employee entity may have Employee_ID, emp_name, salary as attributes.
- In a database management system an attribute is a database component, such as field or column of a table.

Example :

The entity student has attributes like student_id, student_name. In this every attribute has a value. Here 101 is the value for the attribute student_id, Kunal is the value for attribute student_name.

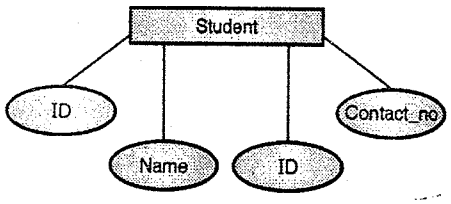


Fig. 1.8.1

There are five different types of attributes in Database Management System :

1. Single-valued Attribute

- A single-valued attribute is the attribute which can hold a single value for the single entity.
- **Example** : In the entity student, student_name is the single-valued attribute since a student have a single value for name attribute.

2. Multi-valued Attribute

- A multi-valued attribute is the attribute which can hold multiple values for the single entity.
- **Example :** In the entity student, the attribute student_contact no could be considered a multi-value attribute since a student could have multiple contact numbers.

3. Simple Attribute

- An attribute whose value cannot be further divided is known as simple attribute. That means it is atomic in nature.
- **Example :** In the entity student, the attribute student_age cannot be divided. Therefore student_age is the simple attribute of student entity.

4. Composite Attribute

- The composite attributes are the attributes which can be further divided into sub parts. These sub parts represent the basic entities with their independent meaning.
- **Example** : In the entity student, student_name is the composite attribute, we can divide this attribute in three different sub parts: First_name, Middle_name and Last_name.

5. Derived Attribute

- The attribute which is not physically exist in database, but its value can be calculated from the other present attributes is known as derived attribute.
- **Example :** In the entity student, we can calculate the average age of students. This average age is not physically present in the database but it can be derived from the attribute student_age;

Syllabus Topic : Relationships

1.8.2.3 Relationships

The association between two different entities is called as relationship. In the real world application, what does one entity do with the other, how do they connect to each other ?

For example : An employee works at a department, a student enrolls for a course. Here, works at and Enrolls are called relationships.

The Degree of Relationships

The degree of relationship refers to number of entities participated in the relation.

1. Unary Relationship

- A unary relationship exists when there is relation between single entity. A unary relationship is also known as recursive relationship in which an entity relates with itself.
- **Example** : A person can be in the relationship with another person, such as :-
- A woman who can be someone's mother
- A person that is a someone's child.

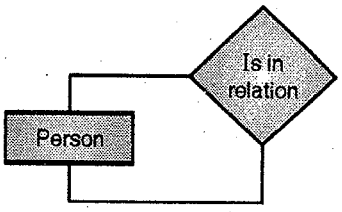


Fig. 1.8.2

2. Binary Relationship

- A binary relationship exist only when there is relation between only two entities. In this case the degree of relation is two.
- **Example :** A teacher teaches student. In this teacher and student are two different entities which are connected with each other via relation Teaches.

Fig. 1.8.3

3. Ternary Relationship

- A ternary relationship exists when there are relations between three entities. In ternary relation the degree of relation is six.
- **Example :** A person can be a student and a person also can be teacher. Here teacher, student and person are three entities which are related to each other.

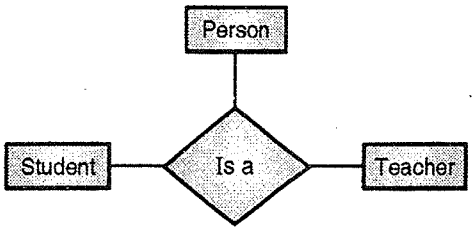


Fig. 1.8.4

4. Quaternary Relationship

- A quaternary relationship exists when there are relations between four entities. In quaternary relation the degree of relation is eight.
- **Example :** The four entities Employee, Management Faculty, Teaching Faculty, and Non-Teaching Faculty are connected with each other via is a relationship.

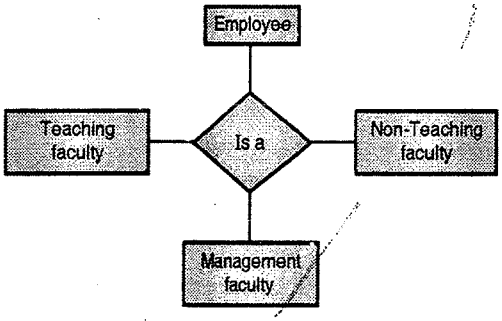


Fig. 1.8.5

Syllabus Topic : Constraints

1.8.2.4 Constraints

There are certain constraints in E-R enterprise schema to which the contents of a database must conform.

Two types of constraints are

1. Mapping cardinalities
2. Participation constraints

1. Mapping Cardinalities SPPU - Dec. 13, May 14

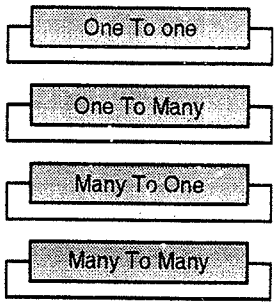
University Questions

Q. What is meant by mapping cardinality? **(Dec. 2013, 2 Marks)**

Q. What is meant by Mapping Cardinality? Explain different types of Cardinalities for a binary relationship with example. **(May 2014, 4 Marks)**

Cardinality in DBMS

- In Database Management System the term 'Cardinality' refers to the uniqueness of data values contained in a column.
- If the column contains a large percentage of totally unique values then it is considered as high cardinality while if the column contains a lot of "repeats" in its data range, it is known as Low cardinality.



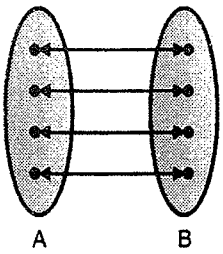
- Sometimes cardinality also refers to the relationships between tables. Cardinality between tables can be one-to-one, many-to-one or many-to-many.
- A relationship where two entities are participating is called a **binary relationship.**

Mapping Cardinalities

- Mapping cardinality expresses the number of entities to which another entity can be associated with in a relationship-set.
- It defines the relationship between two entities via a relationship set. Consider in a binary relationship set R (relation) between entity of set A and set B. For this relationship mapping cardinalities are as Follows :

One to One

- An entity of entity-set A can be associated with at most one entity of entity-set B and vice versa that means an entity in entity-set B can be associated with at most one entity of entity-set A.



(a)

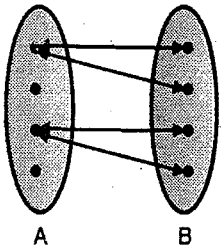
Example : In following example one state have only one capital.

(b)

Fig. 1.8.6

One to Many

- In this type an entity in set A is associated with many other entities in set B.
- But an entity in entity set B can be associated with maximum of one entity in entity set A.



(a)

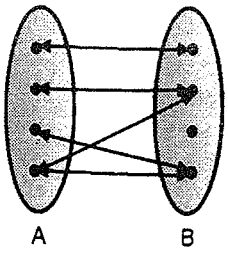
Example : In following example one teacher can teach many students.

(b)

Fig. 1.8.7

Many to One

- In this type an entity in set A is associated with at most one entity in set B. And an entity in set B can be associated with number of entities in set A.



(a)

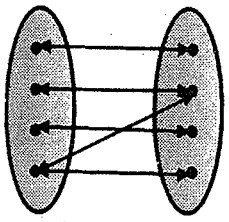
Example : In following example many students can enroll in one school.

(b)

Fig. 1.8.8

Many to Many

- In this type any entity in entity set A is associated with number of entities in entity set B.



(a)

- An entity in entity set B is associated with number of entities in set entity A.
- **Example :** Many students learn many subjects.

(b)

Fig. 1.8.9

The appropriate mapping cardinality for a particular relationship set is depending upon the real world situation to which the relationship set is modeling.

2. Participation Constraints

There are two types of participation constraints.

i. Total participation

ii. Partial participation

i. **Total Participation :** The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.

ii. **Partial Participation :** The participation of an entity set E in a relationship set R is said to be partial if only some entities in E participates in relationships in R.

- For example, in a college database system, consider teachers are assigned as project guides to every student. In this case every Student entity is related with teacher entity through the relationship "Guide".
- Hence participation of student in the relationship guidance is total. But it is not compulsory that every teacher should guide the students.
- Hence it is possible that not all the teacher entities are related with the student through the relationship "Guide". Here the participation of teacher in the "guide" relationship set is partial.

Syllabus Topic : Keys

1.8.2.5 Keys

SPPU - Dec. 16

University Question

Q. Explain the distinction among the terms primary key, candidate key, and super key. **(Dec. 2016, 5 Marks)**

The specification of differentiation of entities in a given entity set is very important. The entities can be differentiated in terms of attributes. Here the values of attributes come in picture. These values should be different to identify the attributes uniquely. It is necessary that in an entity set, no two entities should have same values for all the attributes.

In the database schema the notion of key is directly applies to entity sets. The key for an entity in entity set is an attribute or set of attributes which is used to distinguish entities from each other.

Keys are also used to identify relationships uniquely and differentiate these relationships from each other.

There are six types of keys available in DBMS :

1. Primary Key

- Primary key uniquely identify each entity in the entity set. It must have unique values and cannot hold null values. Let, R be a relationship set having entity sets E1,E2,....En. Consider primary key (Ei) denotes the set of attributes that forms the primary key for entity ser Ei. The set of attributes associated with the relationship set R is responsible for composition of primary key for that relationship set.
- **Example :** In Bank database, the account_number entity should be primary key. Because this field cannot be kept NULL as well as no account_number should be repeated.

2. Super Key

- This key is formed by combining more than one attributes for the purpose of uniquely identifying entities.
- **Example :** In student database having attributes Student_reg_id, Student_roll_no, Sutdent_name, Address, Contact_no.
- The Super keys are :
  {Student_reg_id}
  {Student_roll_no}
  { Student_reg_id, Student_roll_no }
  { Student_reg_id, Sutdent_name }
  { Student_roll_no, Sutdent_name }
  {Student_reg_id, Student_roll_no, Sutdent_name }
- It means super key can be any combination of attributes, so that identifying the record becomes easier.

3. Candidate Key

- Candidate key is formed by collection of attributes which hold unique values. A super key without redundant values is known as candidate key. Candidate keys are selected from the set of super keys.
- Candidate key are also known as minimal super key having uniqueness property. The attribute which do not contain duplicate value, may be a candidate key.

Example : In student database with attributes Student_reg_id, Student_roll_no, Sutdent_name, Address, Contact_no.

- The Candidate keys are :
  {Student_reg_id}
  {Student_roll_no}
  { Student_reg_id, Student_roll_no }
- It means candidate key can be any combination of key attributes, so that identifying the record from the table becomes easier.

4. Alternate Keys

- From all candidate keys, only one key gets selected as primary key, remaining keys are known as alternative or secondary keys.
- **Example :** In student table Student_address, Contact_no, Date_Of_Birth are the alternative keys.

5. Composite Key

- A key which consists of more than one attributes to uniquely identify rows in a table is called composite key. It is also known as compound key.
- **Example :** In student databse having attributes Student_reg_id, Student_roll_no, Sutdent_name, Address, Contact_no.
- The composite keys are :
  { Student_reg_id, Student_roll_no }
  { Student_reg_id, Sutdent_name }
  { Student_roll_no, Sutdent_name }
  { Student_reg_id, Student_roll_no, Sutdent_name }

These sets of keys are used to uniquely identify the record from the table.

Syllabus Topic : Design Process

1.9 Database Design Process

The database design process includes following six stages.

1. Requirement analysis
2. Conceptual database design
3. Choice of the DBMS
4. Logical Database Design
5. Physical design
6. Implementation

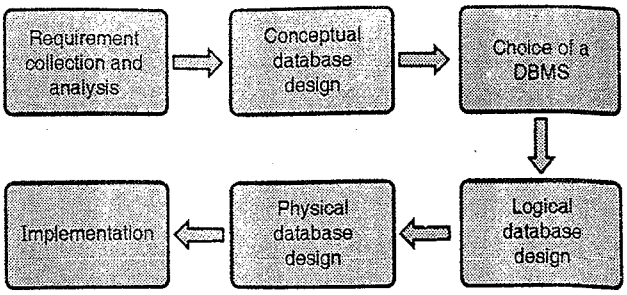


Fig. 1.9.1 : Steps of database design

1. Requirement analysis

- This stage involves the data gathering of requirements from user. Here we discover DATA and OPERATIONS requirements by interaction with the customer.
- The discussion about what current data is and which operations user expect to perform on the database.
- Here we also discuss how data elements are related with each other.

2. Conceptual database design

- The main Purpose of this stage is to produce a conceptual schema of the database using concepts of the high level data model.
- This stage does not include implementation details. It is expected that this design should understood by non-technical users.
- It has detail information about the all the objects of the domain.
- The conceptual database design is independent of the DBMS to be used.
- This design cannot be used directly to implement the database.

3. Choice of the DBMS

- The main purpose of this stage is to select the best suitable database framework for implementing the produced schema.
- The schema includes :
  - Type of DBMS (relational, network, deductive, Object Oriented, ...)
  - User and programmer interfaces
  - Type of query language

- The choice of database is also depends upon technical factors like whether the DBMS has support the required tasks or not.
- The economic factors are also get considered which includes
  - Software and hardware purchase and maintenance.
- Training of staff
- The organisational factors considered are :
  - Platforms supported, availability of vendor services

4. Logical Design

- The main Purpose of logical design is to transform the generic, DBMS independent conceptual schema in the data model of the chosen DBMS. This technique is also called as data model mapping.
- There are two stages of mapping
  1. System independent mapping : In this stage, there is no consideration of characteristics of database i.e. it is totally independent of database system.
  2. Tailoring to DBMS : The same model can be implemented by different DBMS.
- Logical design produces set of DDL statements in the language of the chosen DBMS

5. Physical Design

- The Purpose Physical Design is to select the specific storage structures and access paths for the database files.
- The performances factor is mainly considered.
- Performance of system is based on following factors.
  - **Response time :** The database access time should be less for data items which are frequently used by transactions.
  - **Space utilisation :** Less frequently used data and queries may be archived to save the space.
  - **Transaction throughput :** The number of transactions that can be processed per minute should be more.

6. Implementation

- This is the most important and last stage in design process. In it database is created.
- The DDL statements are compiled and executed.
- In this stage application programs are written with embedded DML statements
- From here operational phase will be initiated.

Syllabus Topic : Entity Relationship Model

1.10 Entity Relationship Model

- In 1976 Entity relationship model was developed. It is useful in conceptual design. It is the high level data model.
- This model defines the data objects and their relationship. It is the popular model in database.
- This model consists of entities and relationships between those entities. An entity is nothing but a thing having its own properties. These properties helps to differentiate the object (entity) from other objects..

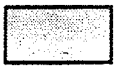
Syllabus Topic : ER Diagram

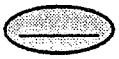
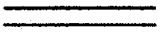
1.11 ER Diagram

The pictorial representation of data using different conventions which state that how these data are related with each other is known as Entity Relationship Diagram. E-R diagrams express the logical structure of database in graphical manner. Special symbols are used to draw an ER-Diagram. Every symbol has its own meaning.

Example of various symbols used in ER Diagram

Following are the symbol / notations used in ER-Diagram :

Symbol	Symbol Name	Symbol Description
Entities		
	Rectangle	**Entity**

Symbol	Symbol Name	Symbol Description
	Double rectangle	**Weak entity**
Attributes		
	Ellipse	**Attribute**
	Ellipse with Under Line	**Key Attribute**
	Double Ellipse	**Multi-valued Attribute**
	Dashed Ellipse	**Derived attribute**
Relationships		
	Diamond	**The relationship**
	Line	**Links attributes to entity sets and entity sets to relationship sets.**
	Double Line	**Represents total participation of an entity in relationship set**

Representations

1. Entity

An **Entity** is any object, place, person or class. In E-R Diagram, an **entity** is represented using rectangles. Consider an example of an Organization. Employee, Manager, Department, Product etc. are considered as entities.

Fig. 1.11.1

Here employee and department are entities.

2. Weak Entity

Weak entity is an entity which depends upon another entity. Weak entity is represented by double rectangle. Subject is the weak entity. Because subject is depends on course.

Fig.1.11.2

3. Attribute

Attributes are nothing but the properties of entity. Here Stud_id, Name and address are attributes of entity Student.

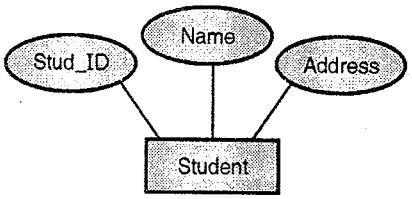


Fig. 1.11.3

4. Key Attribute (Primary key)

To identify attribute uniquely we set the key to the attribute. It is denoted by underline.

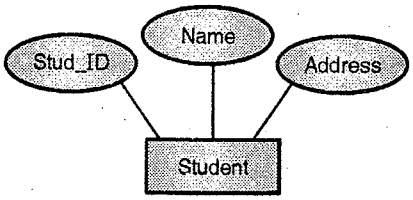


Fig. 1.11.4

5. Multi valued Attribute

The attribute which have multiple values is known as multi valued attribute.

Here Phone No is multi valued attribute as a person can have more than one phone numbers.

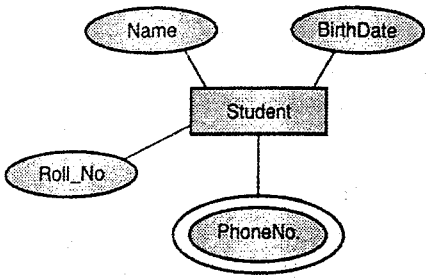


Fig. 1.11.5

6. Derived Attribute

Derived attributes are the attributes that do not exist physically in the database, but their values can be derived from other attributes present in the database.

For example : Age can be derived from data_of_birth.

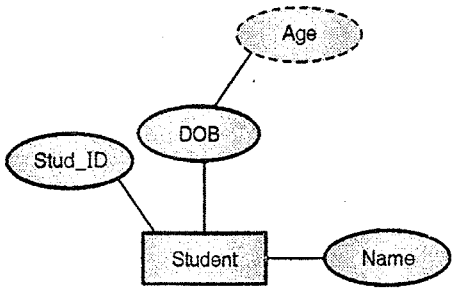


Fig. 1.11.6

7. Relationship

A relationship describes how entities interact with each other. For example, the entity "carpenter" may be related to the entity "table" by the relationship "builds". Relationships are represented by diamond shapes and are labeled using verbs.

Fig. 1.11.7

8. Recursive Relationship

If the same entity participates more than once in a relationship it is known as a recursive relationship. Consider an example where an employee can be a supervisor and be supervised by manager, so there is a recursive relationship.

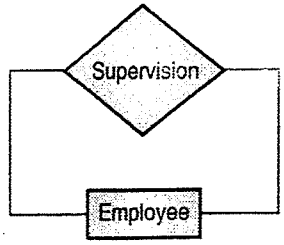


Fig. 1.11.8

Following E-R diagram represent the relationship between two entity sets teacher and student related through binary relationship guide.

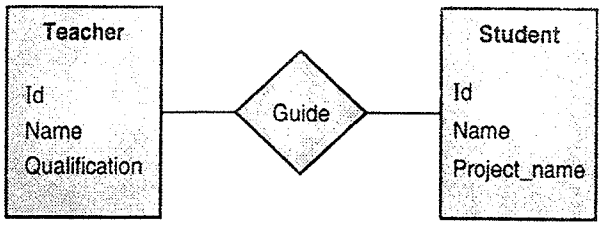


Fig. 1.11.9

The attributes of entity set teacher are

- Id
- Name
- Qualification

The attributes of entity set student are

- Id
- Name
- Project_name

1.11.1 Mapping Cardinality in E-R Diagram

In the two entities Teacher and Student, the relationship *Guide* may be

1. One to One
2. One to Many
3. Many to One
4. Many to Many

1. One to One

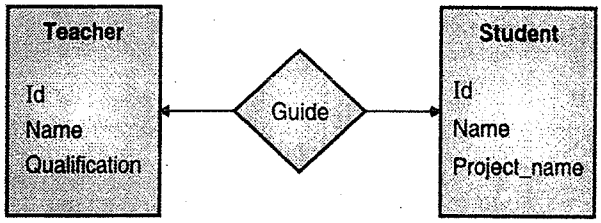


Fig. 1.11.10

In one to one mapping cardinality directed lines from relationship *Guide* are drawn towards both entity sets Teacher and Student. In this, the teacher can guide at most one student and the student can take guidance from at most one teacher.

2. One to Many

In one to many mapping cardinality directed line from relationship *Guide* to entity set Teacher is drawn and undirected line from relationship *Guide* to entity set Student is drawn. In this, the teacher can guide many students but a student can take guidance from at most one teacher.

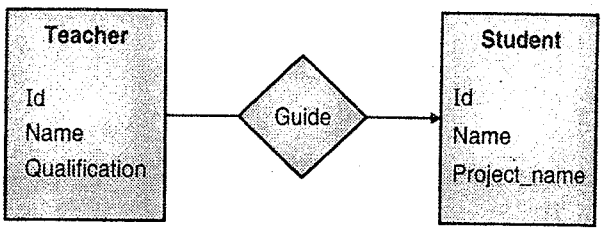


Fig. 1.11.11

3. Many to One

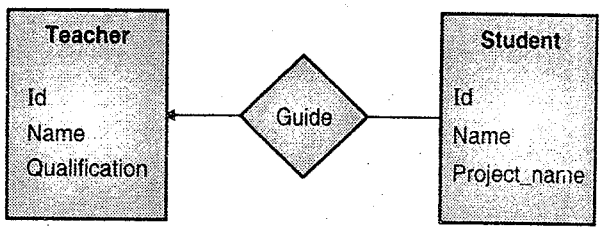


Fig. 1.11.12

In many to one mapping cardinality undirected line from relationship *Guide* to entity set Teacher is drawn and directed line from relationship *Guide* to entity set Student is drawn. In this, the teacher can guide at most one student but a student can take guidance from many teachers.

4. Many to Many

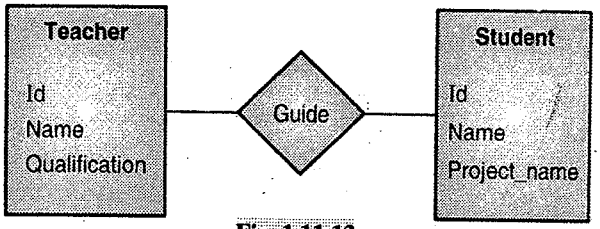


Fig. 1.11.13

In one to one mapping cardinality directed lines from relationship *Guide* are drawn towards both entity sets Teacher and Student. In this, the teacher can guide many students and the student can also take guidance from many teachers.

Points to remember while drawing an ER diagram

1. Initially identify all entities and their relationships with each other in the given database system.
2. No entity should be repeated in a particular diagram.
3. Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Try to give user friendly words while naming. The name should also be meaningful, unique and easily understandable
4. Do not set unclear, redundant or unnecessary relationships between entities.
5. Never connect a relationship to another relationship.
6. Using colors helps to make the diagram easily understandable. It helps in differentiation and classification

1.11.2 Examples of ER Diagram

E-R diagram with multi valued and derived attributes

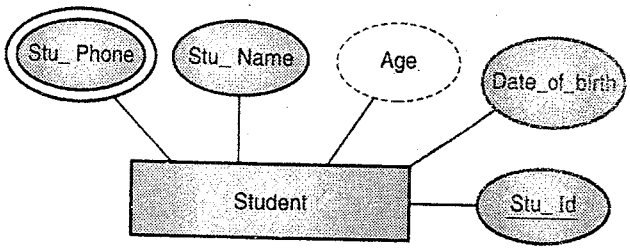


Fig. 1.11.14

Total Participation of an Entity set

If in a relationship set, every entity in entity set has one relationship then it called as total participation of entity set.

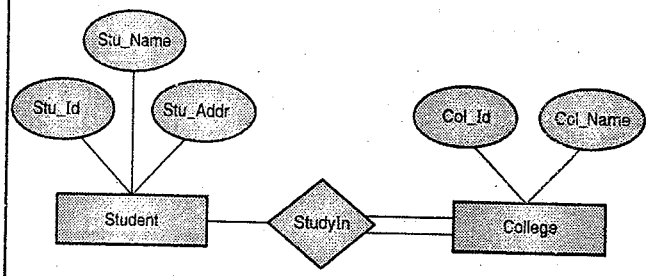


Fig. 1.11.15

In the Fig. 1.11.15, each college must have at least one associated student. A Total participation of an entity set represents that all the entities in the entity set should have minimum one relationship in a relationship set. For example: In the above diagram each college must have at-least one associated Student.

ER diagram for Database of employee

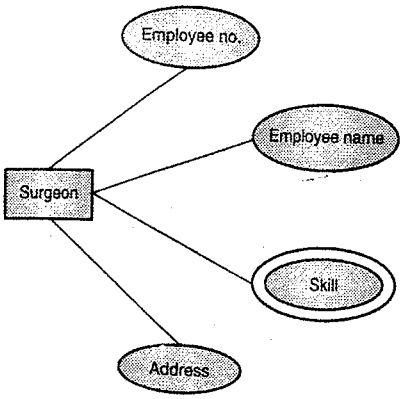


Fig. 1.11.16

Ex. 1.11.1

Following requirements are given for a database of the National Hockey League The NHL has many teams,

Each team has a name, a city, a coach, a captain, and a set of players, Each player belongs to only one team,

Each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records, A team captain is also a player, A game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 1999) and a score (such as 4 to 2). Construct an ER diagram for the NHL database.

Soln. :

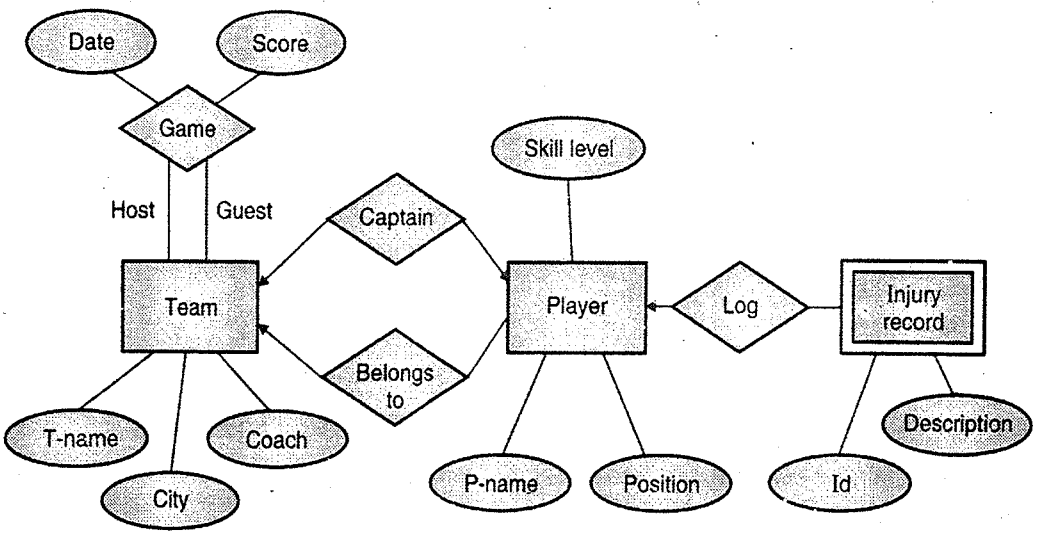


Fig. P. 1.11.1

Ex. 1.11.2 SPPU - Oct. 2016(In sem), 5 Marks

Draw an ER-Diagram for online Book Shop which should consist of entity set, attribute, relationship, mapping cardinality and keys, it will maintain information about all Customers, books, book author, publisher, billing etc.

Soln. :

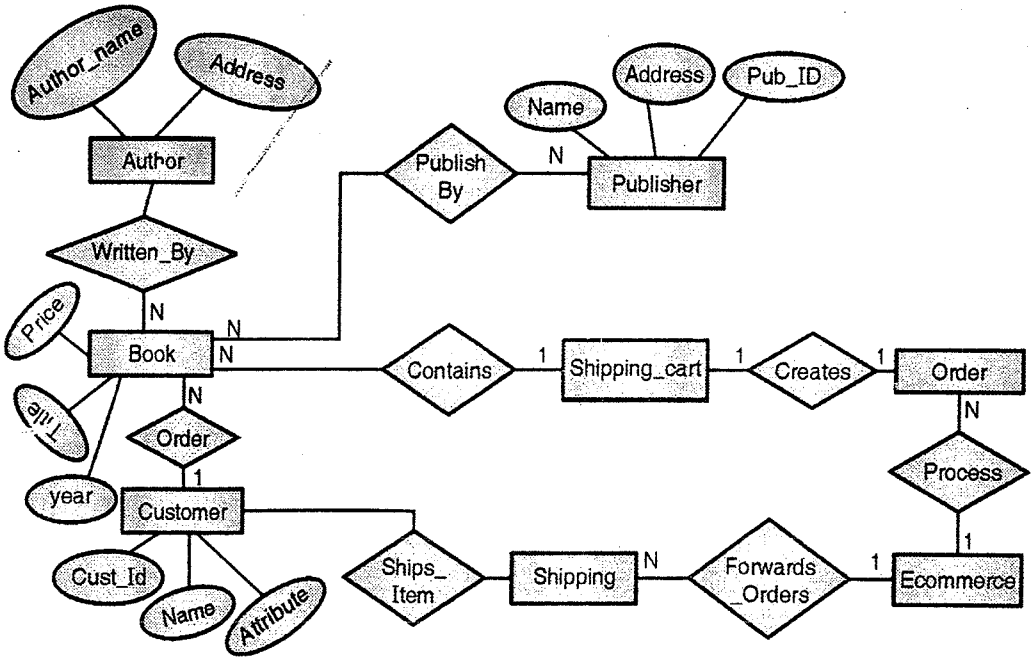


Fig. P. 1.11.2

Ex. 1.11.3

Draw an ER-Diagram for hospital management where patient take treatments from physician and also he/she can claim a medical insurance.

Soln. :

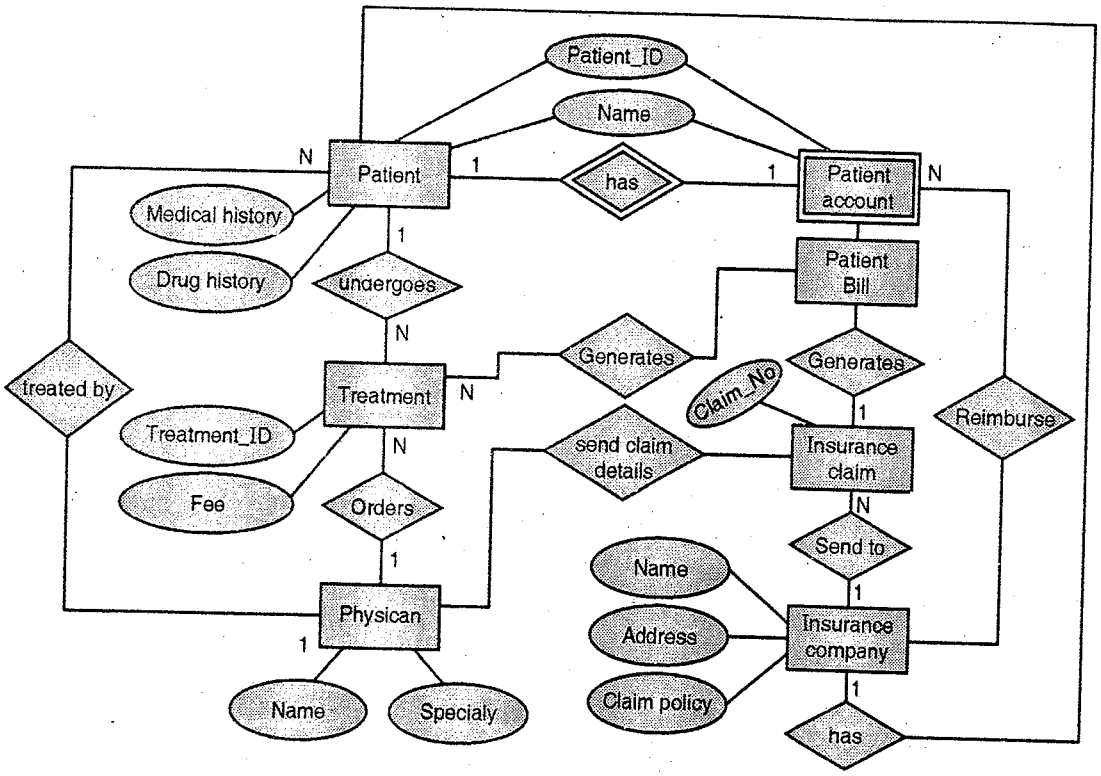


Fig. P. 1.11.3 : ER diagram for Hospital management system

Ex. 1.11.4 :

Draw an ER-Diagram for Online Sales system in which customer can order terms online and pay through credit card.

Soln. :

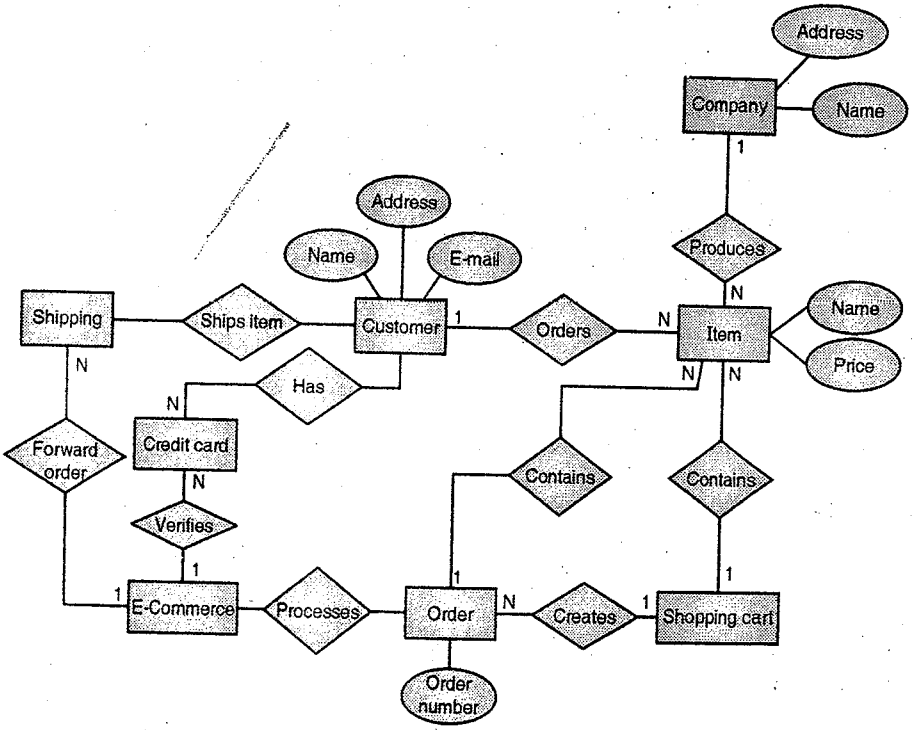


Fig. P. 1.11.5 : ER diagram for Online sales system

Ex. 1.11.5 SPPU - May 2016, 5 Marks

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

Soln. :

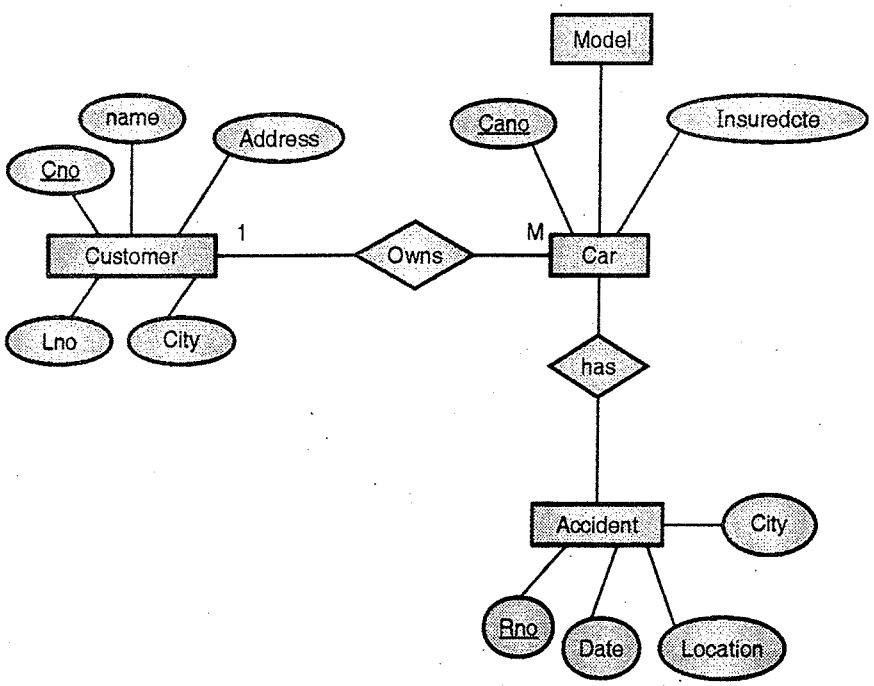


Fig. P.1.11.5

Ex. 1.11.6 :

Following information is maintained for online bookstore

(i) Books(ISBN,price,title,year)

(ii) Author(name,address,url)

(iii) Publisher((name,address,phone,url)

(iv) Customer(name,address,email,phone) (name is discriminating attribute)

(v) Shoppingbasket(basketID)

Construct ER diagram with following constraints :

Each book should have author and publisher. Book may have more than one author. Each customer have a dedicated shopping basket. Books can further be categorized as books, music, cassette or compact disks.

Soln. :

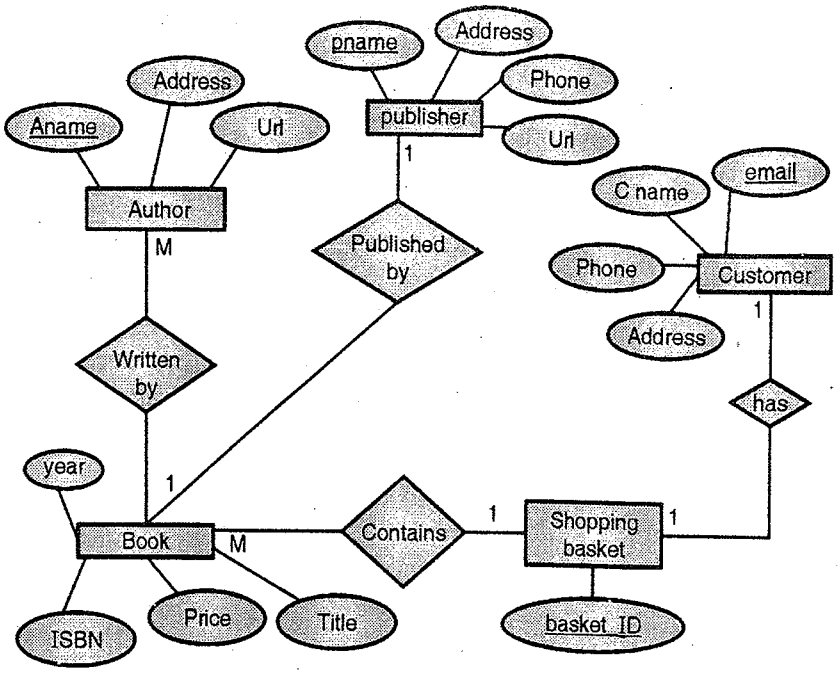


Fig. P.1.11.6

Syllabus Topic : Design Issues

1.12 Design Issues

It is possible to define a set of entities and the relationships among them in a number of different ways. But sometimes it is difficult to decide on the best way to model the application. That means it is hard to find the best option to design. Sometimes it is hard to decide whether something should be represented as an Entity, an attribute or a relationship. These basic issues in the design of an E-R database schema are described as below :

Use of entity sets vs. attributes

- While designing ER-Diagram the question arises as whether a value is represented as a separate entity-set or an attribute?
- The Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.
- Representing a value as a separate entity-set provides a scope to add details later.

For Example

- As shown in Fig. 1.12.1 employee entity has 5 attributes(emp_id, name, phone, city, street).
- An employee has single name but he/she can have multiple phone numbers so it is good to represent phone as a separate entity with two attributes phone_number and location; rather than an attribute. The location stands for office or home. If the phone numbers with different locations are important then we cannot add the phone as attribute, rather we will add it as separate entity having its own attributes.

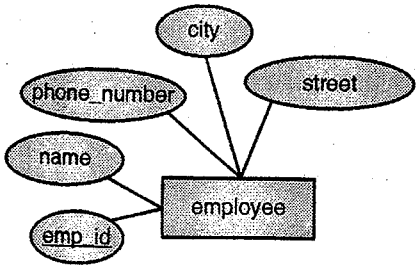


Fig. 1.12.1

- The relationship *emp_phone* can be created between entities employee and phone as follows :

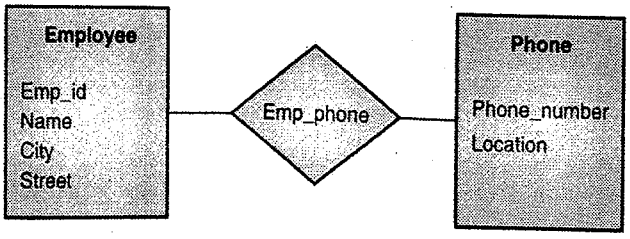


Fig. 1.12.2

- But if the phone number is not so important or detailed concept in the system, then it can be taken as multi-valued attribute.
- Number of times it becomes difficult to set the importance of any element, which makes it complicated to decide whether to make it an attribute or entity.

Use of entity sets vs. relationship sets

- In the situation where ER Diagram is designed to represent loan given to exactly one customer and each loan is given at a particular bank branch.
- We assumed that a loan is modeled as an entity. An alternative is to model a loan as a relationship between customers and branches, with loan-id and amount as descriptive attributes as follows :

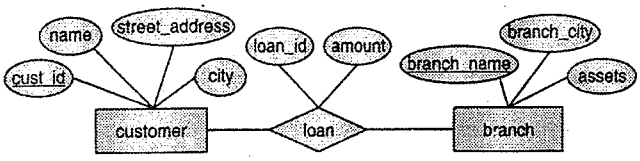


Fig. 1.12.3 : ER diagram for Loan

- Each loan is represented by a relationship between a customer and a branch.
- But in another situation in which several customers hold a joint loan, we must define a separate relationship for each joint-loan-holder.
- So each relationship has the same value for the loan-number attribute and amount attribute. That is more than one customer can hold a loan so the value of loan- id and amount attribute are same for those customers; if for every customer separate relationship is defined then this attribute values are replicated in each relationship.
- This replication causes two problems as :
  1. Waste of storage space
  2. Data in an inconsistent state, if update operation is not well-performed.
- To solve this issue one possible guideline for determining whether to use an entity set or a relation- ship set in ER diagram design, is to choose an entity set to describe an thing and choose a relationship set to describe an action that occurs between entities.
- This approach can also be useful in deciding whether certain attributes may be more appropriately expressed as relationships.

Binary versus n-ary relationship sets

Mostly databases use binary relationships. Some non-binary relationships could actually be better represented by several binary relationships. For example,

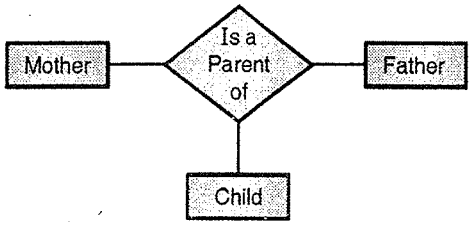


Fig. 1.12.4

Fig. 1.12.4 shows ternary relationship Parent, which relates a Child with his/her Mother and Father. However, it could also be represented by two binary relationships, Mother and Father, relate a Child to his/her Mother and Father separately as follows :

(a)

(b)

Fig. 1.12.5

Advantages of using two relationships Mother and Father allows us to record a child's mother, even if child's father's identity is not known; a null value would be required if the ternary relationship parent is used. This null representation is avoided in binary relationship.

For simplicity, consider the abstract ternary relationship set R, which relates entity sets A, B, and C.

We replace the relationship set R by an entity set E, and create three relationship sets :

- Relationship set RA relates entities E and A.

- Relationship set RB relates entities E and B
- Relationship set RC relates entities E and C

Restricting the E-R model to include only binary relationship sets is not always desirable. It is not possible to translate constraints on the ternary relationship into constraints on the binary relationships.

For example, consider a constraint that says that R is many-to-one relationship from A, B to C; that is, each pair of entities from A and B is associated with at most one in C entity. This constraint cannot be expressed by using cardinality constraints on the relationship sets RA, RB, and RC

Placement of relationship attributes

- Relationship set may have attributes which describe each relation with entities.
- The problem arises where to place this descriptive attribute; whether it is places with one of the entity or it will be placed with the separate relationship-set.

For example

- Consider *loan* relation is related with customer and account entity. If loan relationship has attributes as loan number and amount. We can associate these attributes in the account table or in the loan table.
- The choice of attribute placement is more clear-cut for many-to-many relationship sets. For example: a customer may have one or more number of accounts, and there may be one or more customers for single account.
- To express the date of a specific customer's last account access, then the access date must be an attribute of the depositor relationship set, it should not be one of the participating entities.
- If access-date were an attribute of account, in that case , we could not determine which customer made the most recent access to a joint account.
- When an attribute is determined by the combination of participating entity sets, instead of either entity independently, that attribute must be associated with the many-to-many relationship set.
- In such cases the decision of design: where to place descriptive attributes - as a relationship or entity attribute - should reflect the properties of the business being modeled.
- The cardinality ratio of a relationship can affect the placement of relationship at- tributes. Thus, attributes of one-to-one or one-to-many relationship sets can be associated with one of the participating entity sets, rather than with the relationship set.

Syllabus Topic : Extended ER Features

1.13 Extended ER Features

SPPU - Dec. 13, May 14, Dec. 16

University Questions

Q. Explain the different constraints on specialization/generalization with suitable example. **(Dec. 2013, 4 Marks)**

Q. Explain extended ER features Specialization, Generalization and Aggregation with Example and diagrams. **(May 2014, 8 Marks)**

Q. Explain the concept of specialization & generalization in E-R Model using suitable example. **(Dec. 2016, 5 Marks)**

We can use basic E-R concepts to develop most database features, but to express database deeply some extended features available in ER Model which are known as **Extended ER-Features**. These are as follows :

1.13.1 Specialization

- In an entity set, sometimes further sub grouping is also possible depending upon certain characteristics. For example, a subset may have some attributes which are not shared by other subset in the entity set. In E-R diagram these distinct subsets can be represented by some means.
- Consider an example of entity set person having attributes name, city and street. The entity person may be further classify as follows:
  - customer
  - employee
- Both of these types of person are described by a set of attributes that contains all the attributes of entity set person with some additional attributes of their own.
- For example, in the description of customer entities, we may add new attribute like customer-

id, whereas in the description of employee entities we may add new attributes like employee-id and salary.

- This process of creating subgroups within an entity set is called **specialization**. The specialization of entity person helps us to distinguish whether a person is an employee or customer depending upon attributes.
- An entity set may be specialized by more than one distinguishing characters. In the above example, job performed by an employee may be a distinguishing character from customer. The employee may be further divided as permanent or temporary employee depending upon some characteristics.

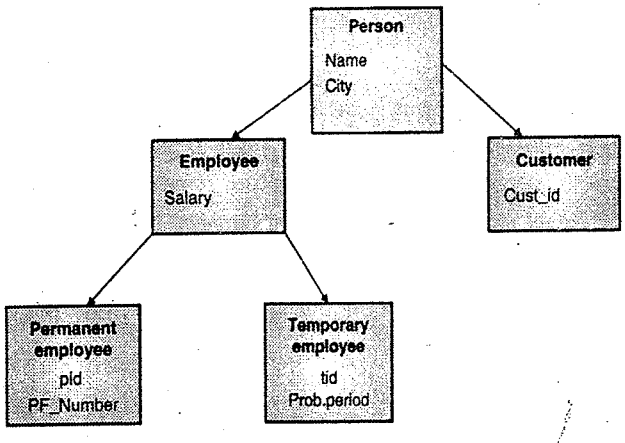


Fig. 1.13.1 : Specialization

1.13.2 Generalization

- In the Fig. 1.13.1, we can observe that the refinement from an initial entity set into subgroups(multiple entity sets) depending upon distinct features shows top-down approch.
- The design process may also proceed into opposite bottom-up approach , in which multiple entity sets are grouped into higher level single entity set depending upon the common characteristics in between these entity sets.
- In out example the entity 'Permananent emplyoee' has following attributes
  - Name
  - City
  - pid
  - Salary
  - Pf_Number
- The entity 'Temporary emplyoee' has following attributes
  - Name
  - City
  - eid
  - Salary
  - Prob_period
- Now if we observe, there are some common attributes between entities 'Permananent emplyoee' and 'Temporary emplyoee'.
- This commonality can be termed as **Generalization** which is containment relationship that exists between higher level entity set and one or more lower level entity set.

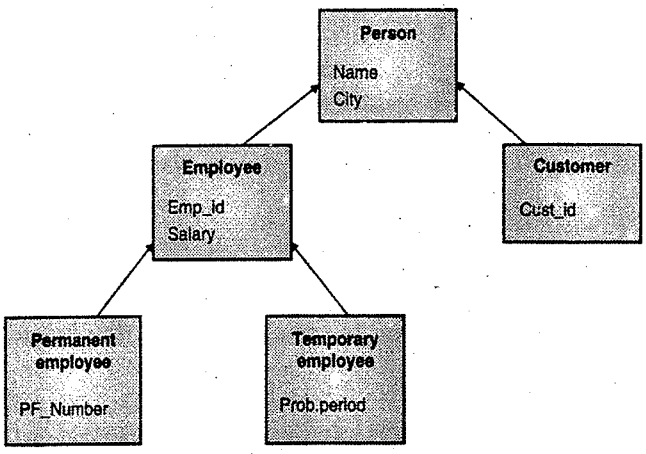


Fig. 1.13.2 : Generalization

- Here the employee is higher level entity set while the entities 'Permananent emplyoee' and 'Temporary emplyoee' are the lower level entity set.
- In this example, the attributes which are conceptually same has different names. e.g. the pid and eid are both nothing but the employee ids of permanent and temporary employees. To create generalization such attributes can be given a common name like emp_id and must be represented in higher level entity *employee*.
- The higher level entity set is also known as Superclass while the lower level entity set also known as subclass.

Attribute Inheritance

- The higher- and lower-level entities created by specialization and generalization has the important property - attribute inheritance
- The attributes of the higher-level entity sets are generally considered to be inherited from the lower-level entity sets.
- In the above example customer and employee both inherit the attributes of person entity set. The description of customer contains name, street, city and additional customer-id attribute. The descriptoin of employee contains name, street, and city and additional employee-id and salary attributes.
- Participation in the relationship sets of higher-level entity is inherted by the lower level entity set.
- Attribute inheritance applies through all tiers of lower-level entity sets. The *employee* and *customer* entity sets can participate in any relationships in which the *person* entity set participates.
- The outcome is basically the same even though the given portion of an E-R model was arrived at by specialization or generalization.
  - A higher-level entity set with attributes and relationships that apply to all of its lower-level entity sets.
  - Lower-level entity sets with unique characters that apply only within a particular lower-level entity set.

Constraints on Generalizations

While designing a database system, the database designer may place certain constraints on a particular generalization.

There are number of such constraints.

(A) One is to detemine which entities can be members of a given lower-level entity set. Such membership may be one of the following :

- **Condition-defined :** In this entity set, it is observed that whether the entity satisfies the specific condition or not.
- Consider an example, assume that the higher-level entity set account contains attribute account-type. Now all account entities are evaluated depending upon the account-type attribute. Only those entities which has account-type as "savings account" are allowed to belong to the lower-level entity set person.
- All entities that has account-type as "current "are included in current account. Here all the lower-level entities are evaluated on the basis of the common attribute as account-type, this type of generalization is called as attribute-defined.
- **User-defined :** This type of lower level entity sets are not constrained by a membership condition. The end user of the database assigns entities to a given entity set. Consider groups are created of students for creating projects. In such case the the teams may be decided by the class teacher.

(B) A second type of constraint check whether entities belong to one or more lower-level entity set within a single generalization.

The lower-level entity sets may be one of the following :

- **Disjoint :** In this generalization , the entity must belong to single lower-level entity set. In the example, the acount_type attribute of account entity satisfies only one condition that it may be saving or current, but not both.
- For a disjointness constraint, it is necessary that an entity should not belong to more than one lower-level entity set.
- **Overlapping :** In this generalizations, One entity may belong to more than one lower-level entity.

Consider an example of IT firm database, where a manager may involve and guide to more than one teams. Hence it appears in more than one lower level team entity sets. Here the generalization is overlapping.

(C) **Completeness :** It specifies whether or not an entity in the higher-level entity set belongs to at least one of the lower-level entity sets within the generalization/ specialization or not.

This constraint may be one of the following :

- **Total generalization or specialization :** Here each higher-level entity must belong to a lower-level entity set.
- **Partial generalization or specialization :** Here higher-level entities may not belong to any lower-level entity set.

1.13.3 Aggregation

In E-R model there is limitation that is it cannot express relationships among relationships. Consider the ternary relationship 'works-on' between a employee, branch, and job. Consider we want to record managers for (employee, branch, job) combinations. An entity set 'manager' is available.

To reprsent this relationship, we can set a quaternary relationships manages between employee, branch, job, and manager. Using the basic E-R modeling constructs, we obtain the following E-R diagram.

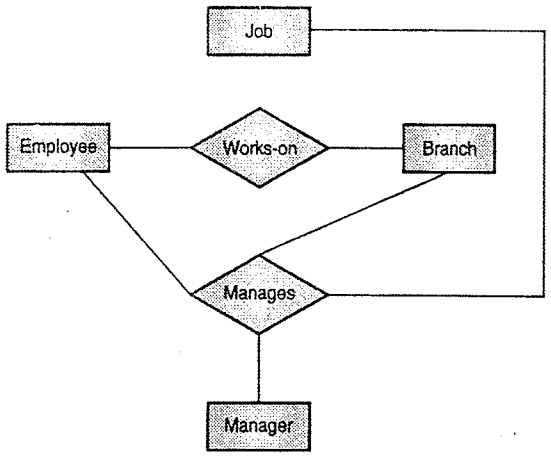


Fig. 1.13.3 : Aggregation

Here the relationship sets 'works-on' and 'manages 'can be combined into one single relationship set. But combining the mis difficult as some employee, branch, job combinations many not have a manager.

Ex. 1.13.1 SPPU - Dec. 2015, 5 Marks

Construct an ER Diagram for a banking Database System. Consider various entities such as Account, Customer, Branch, Loan, Deposit, Borrower, etc. Design Specialization and Generalization EER Feature.

Soln. :

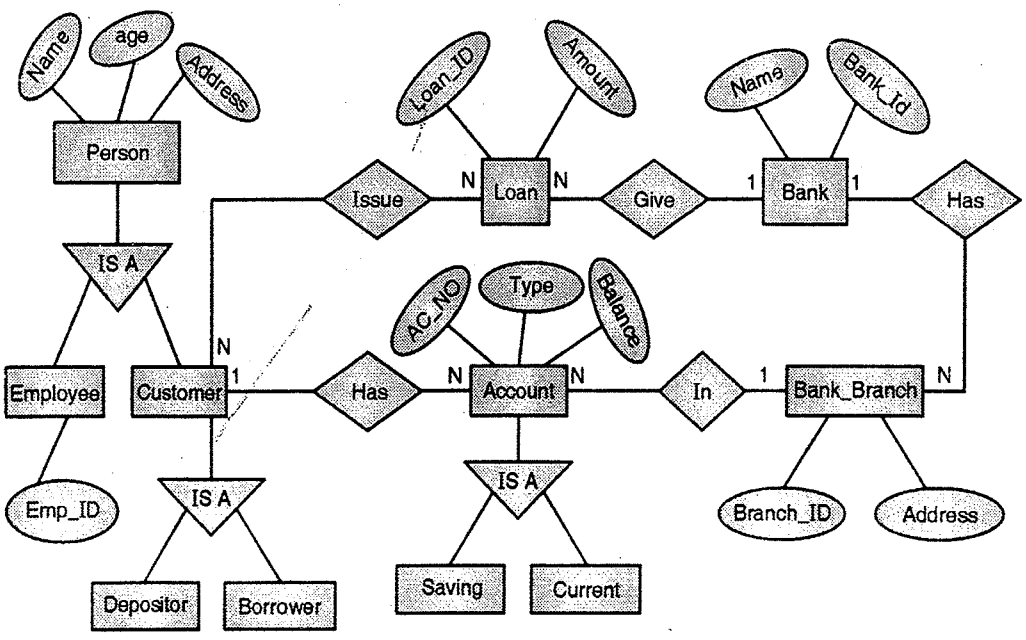


Fig. P. 1.13.1

Ex. 1.13.2 :

Construct an ER Diagram for a Travel Agency. Consider various entities such as travel agency, passenger, Branch, seat, bus, employee, tours etc. Design Specialization and Generalization EER Feature.

Soln. :

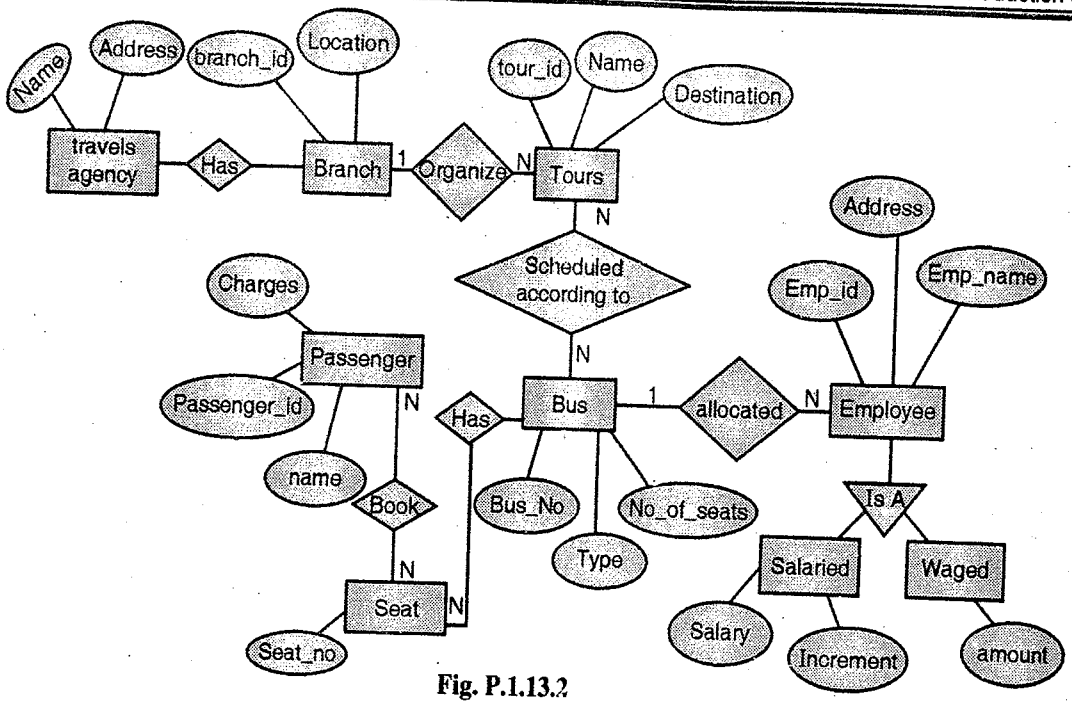


Fig. P.1.13.2

Ex. 1.13.3

For a Library Management System following information is maintained.

Books (Accession_no, Title, Author, Price, Booktype, publisher)

Borrower(Membership_no, Name, Address, Category, max_no_of_books_issued, Accession_no)

Draw E-R Diagram for the above taking into consideration following

Constraints and by making use of at least one Extended ER feature :

i) A book may have more than one author.

ii) There may be more than one copy of a book.

iii) Borrower can be staff or a student. Depending on this category max number of books that can be issued will vary. [i.e. student can ask for max 3 books whereas staff can ask for 10 books]

Soln. :

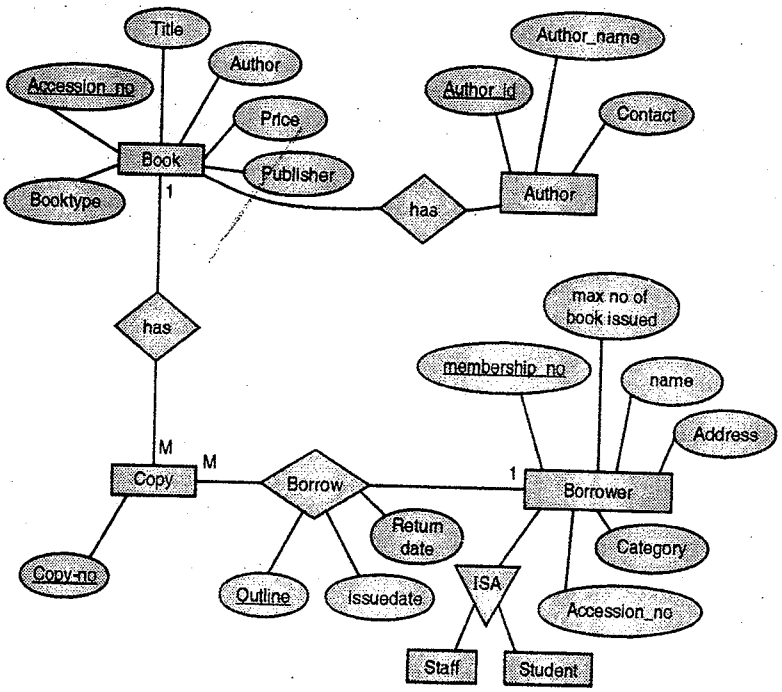


Fig. P.1.13.3 : ER diagram for library management system

Syllabus Topic : Converting ER & EER Diagram into Tables

1.14 Converting ER & EER Diagram into Tables

SPPU - May 13

University Question

Q. Explain with example how E-R diagrams are converted into tables. **(May 2013, 6 Marks)**

As we know ER Diagram gives us good knowledge of entities and their relations. We can understand various mapping cardinalities from ER-Diagram. Using ER Diagram we can easily create Relational Data Model. It is nothing but the logical view of the database. We can convert these ER Diagrams into the tables. There are various steps involved in conversion of ER diagrams into the table.

An ER diagram consists of :

- Entity sets which are represented by rectangular box.
- Relationship sets which are represented by diamond.

We will convert each entity/relationship set into a tables by using following steps:

Consider example see following ER-Diagram which we are going to convert into table using following steps :

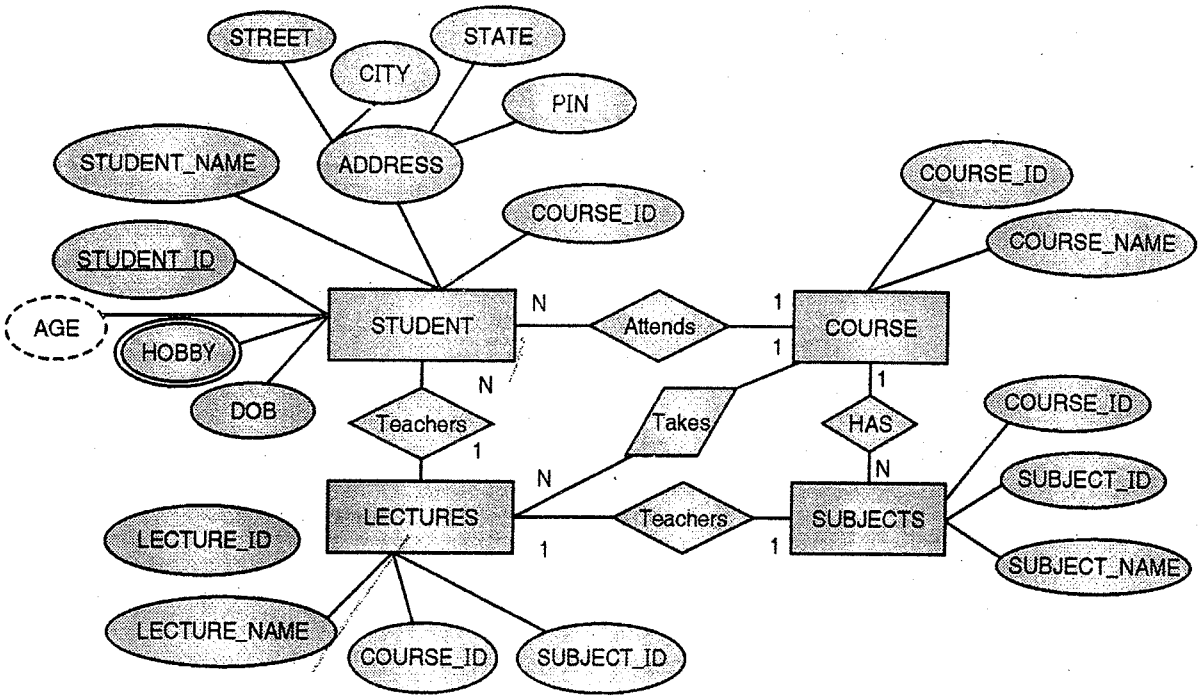


Fig. 1.14.1 : ER diagram

Entity Set ⇒ Table

While converting An ER-Diagram into tables (relational database) first task is to design individual table for each entity in ER diagram. Attributes of entity are treated as fields / columns of tables.

- **Single-valued attribute :** These attributes, whose value is unique, are considered as columns of that table. In the STUDENT Entity, STUDENT_ID, STUDENT_NAME form the columns of STUDENT table. Similarly, LECTURER_ID, LECTURER_NAME form the columns of LECTURER table.
- **Composite Attribute :** Composite attributes are represented in table as individual columns. In above ER diagram, in STUDENT entity ADDRESS is a composite attribute; it is formed by combining STREET, CITY, STATE, and PIN attributes so while converting it into table the table contain columns for STREET, CITY, STATE, and PIN.

- **Multi-valued Attribute :** If any entity has multi-valued attribute then to convert it in table needs to form an individual table for that attribute. In above diagram, in the Student table , hobby attribute is multi-valued. A student can have more than one hobby. So it is difficult to represent multiple values in a single column of STUDENT table. It must be stored separately, so that it is possible to store multiple hobbies. Redundancy in the system should not be created by adding/ removing / deleting hobbies. A separate table STUD_HOBBY with STUDENT_ID and HOBBY as its columns can be created. We can create a composite key using both the columns.
- **Primary key :** In above diagram, STUDENT_ID, LECTURER_ID, COURSE_ID and SUBJECT_ID are the key attributes of the entities. Hence they are the primary keys of respective table.
- **Weak entity :** In above ER diagram no weak entity is present.
- As shown in Fig. 1.14.1 ER-Diagram, it contains 4 entities (STUDENT, COURSE, LECTURER and SUBJECT) which are become independent tables as follows :

Table 1.14.1 : Student

Student_Id	Student_Name	Street	City	State	Pin	Course_Id	Dob

Table 1.14.2 : Stud_Hobby

Student_Id	Hobby

Table 1.14.3 : Lecturer

Lecturer_Id	Lecturer_Name	Course_Id	Subject_Id

Table 1.14.4 : Course

Course_Id	Course_Name

Table 1.14.5 : Subject

Subject_Id	Subject_Name	Course_Id

❑❑❑

CHAPTER 2

UNIT II

SQL and PL/SQL

Syllabus

- **SQL :** Characteristics and advantages
- SQL Data Types and Literals
- DDL, DML, DCL, TCL
- SQL Operators
- Tables : Creating, Modifying, Deleting
- Views : Creating, Dropping, Updating using Views
- Indexes
- SQL DML Queries: SELECT Query and clauses
- Set Operations
- Predicates and Joins
- Set membership
- Tuple Variables
- Set comparison
- Ordering of Tuples,
- Aggregate Functions
- Nested Queries
- Database Modification using SQL Insert, Update and Delete Queries.
- **PL/SQL:** concept of Stored Procedures & Functions
- Cursors, Triggers
- Assertions
- Roles and privileges
- Embedded SQL
- Dynamic SQL

Syllabus Topic : SQL

2.1 SQL - Characteristics and Advantages

- SQL stands for Structured Query Language. SQL is used to communicate with a database. It is the standard language for relational database management systems. SQL statements are used to perform different operations on database like retrieval, insertion, updation and deletion of data.
- SQL is used in various advanced Relational Database Management Systems (RDBMS). Some common RDBMS that use SQL are : MySQL, Oracle, Microsoft Access, Microsoft SQL Server, Sybase, Ingres, etc.
- SQL is developed by IBM as a part of System R project in 1970. Initially it was called as Sequel. SQL was one of the first commercial languages for Edgar F. Codd's relational model. SQL became a standard of the American National Standards Institute (ANSI) in 1986, SQL is a declarative language in which the desired result is given without the specific details about how to accomplish the task.
- The steps required to execute SQL statements are handled transparently by the SQL database. SQL can be characterized as non-procedural because in procedural languages the details of the operations to be specified, such as opening and closing tables, loading and searching indexes, or flushing buffers and writing data to file systems are required which is not necessary in SQL.

Syllabus Topic : Characteristics of SQL

2.1.1 Characteristics of SQL

- SQL is an ANSI and ISO standard computer language for creating and manipulating databases.
- SQL allows the user to create, update, delete, and retrieve data from a database.
- The tokens and syntax of SQL are oriented from English common speech to keep the access barrier as small as possible. Hence it is very simple and easy to learn.
- All the keywords of SQL can be expressed in any combination of upper and lower case characters. It makes no difference whether UPDATE, update, Update, UpDate i.e. the keywords are case insensitive
- SQL is a declarative language, not a procedural one.
- SQL is very powerful language.
- SQL works with database programs like DB2, Oracle, MS Access, Sybase, MS SQL Sever etc.

Syllabus Topic : Advantages of SQL

2.1.2 Advantages of SQL

There are various advantages of SQL

- **High Speed :** SQL Queries can be used to retrieve large amounts of records quickly and efficiently from a database.
- **Portable :** SQL can be run on any platform. Also it can be executed on PCs, laptops, servers and even mobile phones. It runs in local systems, intranet and internet. Databases using SQL can be moved from a device to another without any problems.
- **Well Defined Standards Exist :** SQL databases use long-established standard, which is being adopted by ANSI & ISO. Whereas Non-SQL databases do not adhere to any clear standard.
- **Supports object based programming :** SQL supports various object oriented programming concepts which makes it powerful.
- **Used with all DBMS systems with any vendor :** SQL is used by all the vendors who develop DBMS.
- **No Coding Required :** Using standard SQL it is easier to manage database systems without writing large amount of code.
- **Used for relational databases :** SQL is widely used for number of relational databases.
- **Easy to learn and understand :** SQL mainly consists of English words and hence it is easy to learn and understand the SQL queries.
- **Complete language for a database :** SQL is used to create databases and manage the databases in all aspects.
- **Dynamic database language :** SQL can change the database dynamically at runtime even while the database is being used by users.
- **Can be used as programming and interactive language :** SQL can do both the jobs of being a programming as well as an interactive language at the same time.
- **Client/Server language :** SQL can be used in client server architecture as a mediator between client application and server database.
- **Multiple data views :** We can provide different views(presentations) of contents of a database to different users.
- **Used in internet :** SQL can be used in internet to access the web related data.

Syllabus Topic : SQL Data Types

2.2 SQL Data Types and Literals

2.2.1 SQL Data Types

In SQL, we store the data in tabular format where table (relation) is the combination of rows (tuples) and columns (fields). While creating table we have to assign data types to the columns. These data types are used to decide that which type of data the columns can store.

There are various types of data types in SQL to store different types of data items.

1. CHAR
2. VARCHAR
3. BOOLEAN
4. SMALLINT
5. INTEGER or INT
6. DECIMAL [(p[,s])] or DEC [(p[,s])]
7. NUMERIC [(p,[s])]
8. REAL
9. FLOAT(p)
10. DATE
11. TIME
12. TIMESTAMP
13. CLOB [(length)] or CHARACTER LARGE OBJECT [(length)] or CHAR LARGE OBJECT [(length)]
14. BLOB [(length)] or BINARY LARGE OBJECT [(length)]

(1) CHAR (length)

- The CHAR data type accepts character OR string type of data including Unicode. It is known as fixed length data type. The length of the character string is specified while assigning the data type. For example, CHARACTER(n) where n represents the maximum size of the character string. If size is not specified then the default size will be 1.
- The minimum length of the CHARACTER data type is 1 and maximum length is up to the table page size. Character strings which are larger than the page size of the table can be stored as a Character Large Object (CLOB).
- If value having lower size than the size of CHAR data type is stored in it, then the remaining space is filled with blanks characters. That means it gets wasted.
- If value having greater size than the size of CHAR data type is tried to store, then the extra characters are truncated.

Examples :

CHAR(20) or CHARACTER(20)

'Phoenix', 'INFOTECH'

(2) VARCHAR (length)

- The VARCHAR data type accepts character OR string type of data including Unicode. It is known as variable length data type.
- The length of the character string is specified while assigning the data type which indicates the maximum number of characters it can accept.
- We can assign the length from 1 to the current table page size.
- If value having lower size than the size of VARCHAR data type is stored in it, then the remaining space will get reutilized. That means the memory does not get wasted.
- If you need to store character strings that are longer than the current table page size, the Character Large Object (CLOB) data type should be used.

Examples : VARCHAR(10)

'Phoenix', 'INFOTECH'

(3) BOOLEAN

- The BOOLEAN data type can accepts value either TRUE or FALSE. No need to declare size while declaring the BOOLEAN data type.
- TRUE or FALSE are case insensitive. If you attempt to assign any other value to a BOOLEAN data type, an error gets raised.

Examples : TRUE, true, True, False

(4) SMALLINT

- The SMALLINT data type is used to accept numeric values with default scale as zero. It stores any integer value between the range 2^ -15 and 2^15 -1. Attempting to assign values outside this range causes an error.
- If you assign a numeric value with a precision and scale to a SMALLINT data type, the scale portion truncates, without rounding.

Examples : SMALLINT

-32768, 0, –13.7 (digits to the right of the decimal point are truncated), 32767

(5) INTEGER or INT

- The INTEGER data type is used to accept numeric values with a default scale as zero. It stores any integer value between the range 2^ -31 and 2^31 -1. Attempting to assign values outside this range causes an error.
- If you assign a numeric value with a precision and scale to an INTEGER data type, the scale portion truncates, without rounding.

Examples

– 345, 0, 4532.98 (digits to the right of the decimal point are truncated), 167

(6) DECIMAL [(p[,s])] or DEC [(p[,s])]

- The DECIMAL data type is used to accept floating point values for which you define a precision and a scale in the data type declaration. The precision is a positive integer that represents the total number of digits that the number will contain (precision + scale).
- The scale is a positive integer that represents the number of digits of decimal places which will occur to the right of the decimal point. The scale for a DECIMAL cannot be larger than the precision.
- If you exceed the number of digits expected to the left of the decimal point, an error is thrown. If you exceed the number of expected digits to the right of the decimal point, the extra digits are truncated.

Examples : DECIMAL (10, 3)

98789, 765.123, 10.1234(Final digit is truncated), -987, -897.786, -1234567.1234 (Final digit is truncated)

(7) NUMERIC [(p[,s])]

It is same as of Decimal

(8) FLOAT (p)

The FLOAT data type accepts approximate numeric values, for which you may define a precision up to a maximum of 64. The default precision is 64 if not declared.

Examples : FLOAT(8)

12345678, 1.2, 123.45678, -12345678, -1.2, -123.45678

(9) DATE

- The DATE data type accepts date type of values. No need to assign size while declaring a DATE data type. Date values should be specified in the form: YYYY-MM-DD.
- The value of month must be between 1 and 12, value of day should be between 1 and 31 depending on the month and value of year should be between 0 and 9999. The values should be enclosed in single quotes, preceded by the keyword DATE.

Examples : DATE '1999-01-01'

DATE '2000-2-2'

(10) TIME

- The TIME data type accepts time values. No parameters are required when declaring a TIME data type. The format is : HH:MM:SS. The fractional value can be used to represent nanoseconds.
- The minutes and seconds values must be two digits. Hour values should be between zero 0 and 23, minute values should be between 00 and 59 and second values should be between 00 and 61.999999.
- Values assigned to the TIME data type should be enclosed in single quotes, preceded by keyword TIME.

Examples : TIME '1:10:20'

Syllabus Topic : SQL Literals

2.2.2 SQL Literals

The literal is the constant or fixed data value. For example 'Phoenix', 'Institute' are string literals while 100, 5 are numerical literals and so on. The String, date and time literals are always enclosed in single quotation marks while the numerical literals are without quotation marks. Literal are case sensitive.

SQL Supports following types of literals.

(1) Numeric Literals
(2) Character Literals
(3) String Literals

(4) Date Literals
(5) Time Literals

(1) Numeric Literals

Numeric literals are the sequence of digits proceeded by an optional sign (+ / –) and with an optional decimal point. The Numeric Literals are further classified into two categories :

Integer Literals : These are the whole numbers assigned as data values. An integer can store a maximum of 38 digits of precision.

For example : 100, + 7, – 8 etc.

Number or Floating Point Literals : These are the numbers with decimal point assigned as data values. For example : 10.2, +7.3, -8.2 etc.

(2) Character Literals

Character literal contains single character enclosed in single quotation marks.

For example : 'A', '%', '9', 'z', '('

(3) String Literals

- These are the sequence of characters enclosed in single quotes. In SQL versions up to 7.0, the maximum length of a string literal is 1024. From version 7.1, there is no specific maximum limit on number of characters.
- The character string literals cannot be enclosed in double quotes because it is reserved for delimiting identifiers such as field or table names.

For example : 'Phoenix', 'I' etc.

(4) Date Literals

- These literal are the date type of values in the ANSI date format 'YYYY-MM-DD' or the default date format specified in the application via the 'set date format' operation.
- The date literal is enclosed in single quotation marks.

For Example : '2017-03-18'

(5) Time Literals

- These literal are the Time type of values in the format 'HH:MM' or 'HH:MM:SS'. In addition 'am' or 'pm' can be included at the end for 12-hour time format.
- If the am/pm indicator is excluded, it is assumed that the time is in 24-hour format.
- The date literal is enclosed in single quotation marks.

'11:15', '8:30 am', '06:25:15 pm', '17:00'

Syllabus Topic : DDL, DML, DCL, TCL

2.3 DDL, DML, DCL, TCL

SPPU - May 13, May 14

University Questions

Q. Explain various database Languages. **(May 2013, 8 Marks)**

Q. Write short note on DDL, DML and DCL. **(May 2014, 4 Marks)**

The database languages are categorized as DDL, DML, DCL and TCL.

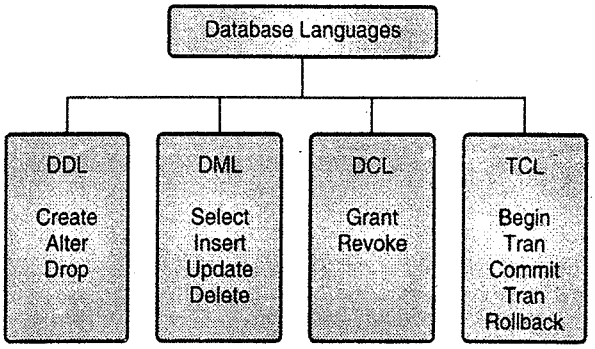


Fig. 2.3.1

2.3.1 Data Definition Language (DDL)

- This language allows the users to define data and their relationship to other types of data. It is used to create data tables, dictionaries, and files within databases.
- The DDL is also used to specify the structure of each table, set of associated values with each attribute, integrity constraints, security and authorization information for all the tables and physical storage structure of all the tables on the disk.
- Let's take SQL for instance to categorize the statements that comes under DDL.
  - To create the database instance – **CREATE**
  - To alter the structure of database – **ALTER**
  - To drop database instances – **DROP**
  - To rename database instances – **RENAME**

2.3.2 Data Manipulation Language (DML)

- The Data Manipulation Language (DML) is used for accessing and manipulating data in a database. DML provides a set of functionalities to support the basic data manipulation operations on the data stored in the database.
- It allows users to access, insert, update, and delete data from the database.
  - To access or read records from table – **SELECT**
  - To insert record into the table – **INSERT**
  - Update the records in table – **UPDATE**
  - Delete the records from the table – **DELETE**

2.3.3 Data Control Language (DCL)

- Data Control Language (DCL) is used to control the user access to the database related elements like tables, views, functions, procedures and packages.
- It provides different levels of access to the objects in the database.
  - To grant access to user – **GRANT**
  - To revoke access from user – **REVOKE**

Grant : GRANT is used to provide the privileges to the users on the database objects. The privileges could be select, delete, update and insert on the tables and views. On the procedures, functions and packages it gives select and execute privileges.

Revoke : REVOKE removes the privileges given on the database objects. All the privileges can be removed at a time or one or more privileges can also be removed from the objects as per requirement.

2.3.4 Transaction Control Language (TCL)

TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.

- **BEGIN Transaction** – opens a transaction
- **COMMIT Transaction** – commits (Save permanently) transactions
- **ROLLBACK Transaction** – ROLLBACK (Cancels. undo) transactions in case of any issue

Syllabus Topic : SQL Operators

2.4 SQL Operators

- An operator is a character or reserved word used in SQL statements to perform different operations like arithmetic or comparison.
- Operators are used to specify conditions in an SQL statement and also used to integrate multiple conditions in SQL statement.
  - Arithmetic operators
  - Comparison operators
  - Logical operators
  - Operators used to negate conditions

2.4.1 Arithmetic Operators

Consider two operands x and y where value of x is 10 while y is 5.

Operators	Descriptions	Examples
+ (Add)	It adds the operands	x + y = 15
- (Subtract)	It subtracts right hand operand from left hand operand	x – y = 5
* (Multiply)	It multiply both operands	x * y = 50
/ (Divide)	It divides left hand operand by right hand operand	x / y = 2
% (Modulo)	It divides left hand operand by right hand operand and returns reminder	x % y = 0

2.4.2 Comparison Operator

Consider two operands x and y where value of **x** is **10** while **y** is **5**.

Operators	Descriptions	Examples
=	Check whether both the operands have same values, if yes condition becomes true.	x=y is not true
>	Check whether left operand is greater than right, if yes condition becomes true	x>y is true
<	Check whether left operand is less than right, if yes condition becomes true	x<y is not true

Operators	Descriptions	Examples
>=	Check whether left operand is greater than or equal to the right operand or not, if yes condition become true	x>=y is true
<=	Check whether left operand is less than or equal to the right operand or not, if yes condition become true	x<=y is not true
!=	Check whether both the operands have same values or not, if not condition become true.	x != y is true
<>	Check whether both the operands have same values or not, if not condition become true.	x<>y is true
!>	Check whether left operand is not greater than the value of right operand	x!>=y is not true
!<	Check whether left operand is not less than the right operand value	x!<=y is true

2.4.3 Logical Operators

Operator	Description
ALL	TRUE if all of a set of comparisons are TRUE.
AND	TRUE if both Boolean expressions are TRUE.
ANY	TRUE if any one of a set of comparisons is TRUE.
BETWEEN	TRUE if the operand is within a range.
EXISTS	TRUE if a sub-query contains any rows.
IN	TRUE if the operand is equal to one of a list of expressions.
LIKE	TRUE if the operand matches a pattern.
NOT	Reverses the value of any other Boolean operator.
OR	TRUE if either Boolean expression is TRUE.
SOME	TRUE if some of a set of comparisons are TRUE.

2.4.4 Bitwise Operators

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise excusive OR

2.4.5 Compound Operators

Operator	Description
+=	Add equals
-=	Subtract equals
*=	Multiply equals
/=	Divide equals
%=	Modulo equals
&=	Bitwise AND equals
^-=	Bitwise exclusive equals
|*=	Bitwise OR equals

Syllabus Topic : Tables - Creating, Modifying, Deleting

Note : All the queries are implemented considering MySQL.

2.5 Tables : Creating, Modifying, Deleting

In DBMS the standard format of storing the data is table. Table is also known as relation. It is the combination of rows and columns.

2.5.1 Creating Table

The **CREATE TABLE** statement is used to create table in database.

Syntax

```
CREATE TABLE table_name (
   column1 datatype[size],
   column2 datatype [size],
   column3 datatype[size],
   ....
);
```

In the CREATE TABLE statement the column parameters specify the names of the columns or fields of the table. The data type is the type of data which we want to store in the respective fields. The fields can hold data of different types like char, varchar, number, date etc.

The optional size value can also be mentioned after the data type. This size value indicated the maximum length of data for the field. If size is not given then default value depending upon the data type is assigned.

Example

Student Table

```
CREATE Table student
(roll_no integer(3), stud_name varchar(20),
bdate date , marks integer(3));
```

In the above example, a table student will be created with following attributes. The roll_no field will contain numerical value of maximum digit 3. The stud_name field will contain string value of maximum length 20. The date field will contain date type value of standard date length. For date data type no need to mention size. The marks field will contain numerical value of maximum digit 3.

The structure of the table will be as follows after record insertion.

Table 2.5.1 : Student

roll_no	stud_name	bdate	marks
101	Kunal	12-02-2000	90
102	Jay	07-08-1999	68
103	Radhika	05-04-2000	85
104	Sagar	13-02-2000	70
105	Supriya	11-08-1999	72

Employee Table

```
CREATE Table employee
(emp_id integer(3), emp_name varchar(20),
Salary integer(7), department varchar(20));
```

Product Table

```
CREATE Table product
(prod_code integer(3), prod_name varchar(20),
price integer(7), category varchar(20));
```

Customer Table

```
CREATE Table customer
(cust_id integer(3), cust_name varchar(20),
address varchar(20), email_id varchar(20));
```

2.5.1.1 Creating New Table from Existing Table

AS SELECT clause is used to create table from existing table. It can be considered as copy of existing table. While creating such table, we have option whether to take all records, fields from existing table or not. We can copy just structure of the existing table also. The different ways of coping table are as given below :

Consider the existing table student as shown in Table 2.5.1. Creating new table same as of existing table.

Syntax

```
Create table table_name
as select * from existing_table_name;
```

For Example

```
Create table newstudent1
as select * from student ;
```

Output : The newly created table will be

Table 2.5.2 : Newstudent1

roll_no	stud_name	bdate	marks
101	Kunal	12-02-2000	90
102	Jay	07-08-1999	68
103	Radhika	05-04-2000	85
104	Sagar	13-02-2000	70
105	Supriya	11-08-1999	72

Creating new table having specific fields but all the records from existing table.

Syntax

```
Create table table_name
as select field_1,field_2... from
existing_table_name;
```

For Example

```
Create table newstudent2
as select roll_no,stud_name from student;
```

Output : The newly created table will be

Table 2.5.3 : Newstudent2

roll_no	stud_name
101	Kunal
102	Jay
103	Radhika
104	Sagar
105	Supriya

Creating new table having specific records but all the fields from existing table.

Syntax

```
Create table table_name
as select * from existing_table_name
where condition
```

For Example

```
Create table newstudent3
as select * from student
where marks > 80;
```

The newly created table will be structure wise same as of existing table, but it will contain records of only those students who got marks above 80.

Output

Table 2.5.4 : Newstudent3

roll_no	stud_name	Bdate	marks
101	Kunal	12-02-2000	90
103	Radhika	05-04-2000	85

Creating new table having no records but all the fields from existing table. That means copying only structure of existing table.

Syntax

```
Create table table_name
 as select * from existing_table_name
where false condition
```

For Example

```
Create table newstudent4
as select * from student
where 1=2;
```

Here 1=2 is the false condition. The newly created table will be structure wise same as of existing table, but it will no records get copied.

Output

Table 2.5.5 : Newstudent4

roll_no	stud_name	bdate	marks

2.5.2 Modifying Table

ALTER TABLE query is used to modify structure of a table. We can add, delete or modify column.

Adding New Column in a Table

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

For Example : Adding column grade in the table student.

```
ALTER TABLE student
ADD Grade varchar(2);
```

Dropping Column from Table

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

For Example : Deleting column grade from the table student.

```
ALTER TABLE student
DROP column Grade;
```

Modifying Column of a Table

```
ALTER TABLE table_name
MODIFY COLUMN column_name data_type;
```

For Example : Changing the data type and size of column roll_number of student table.

```
ALTER TABLE student
modify column roll_no varchar(4);
```

Here we are changing the.

Deleting all the records from Table

Syntax

```
TRUNCATE TABLE table_name;
```

For Example : Deleting all the records from newstudent1

```
TRUNCATE TABLE newstudent1;
```

2.5.3 Deleting Table

DROP TABLE query is used to delete table permanently from the database.

Syntax

```
drop table table_name;
```

For Example : Deleting the newstudent1 table from the database.

```
Drop table newstudent1;
```

Syllabus Topic : Views - Creating, Dropping, Updating View

2.6 View : Creating, Dropping, Updating View

SPPU - May 15

University Question

Q. Explain view objects in SQL with example. **(May 2015, 3 Marks)**

View : In SQL, a view is a virtual table containing the records of one or more tables based on SQL statement executed. Just like a real table, view contains rows and columns. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table. The changes made in a table get automatically reflected in original table and vice versa.

Purpose of View : View is very useful in maintaining the security of database. Consider a base table employee having following data.

Emp_id	emp_name	Salary	Address
E1	Kunal	8000	Camp
E2	Jay	7000	Tilak Road
E3	Radhika	9000	Somwar Peth
E4	Sagar	7800	Warje
E5	Supriya	6700	LS Road

1. Now just consider we want to give this table to any user but don't want to show him salaries of all the employees. In that we can create view from this table which will contain only the part of base table which we wish to show to the user.

   See the following View.

Emp_id	emp_name	Address
E1	Kunal	Camp
E2	Jay	Tilak Road
E3	Radhika	Somwar Peth
E4	Sagar	Warje
E5	Supriya	LS Road

2. Also in multiuser system, it may be possible that more than one user may want to update the data of same table. Consider two users A and B want to update the employee table. In such case we can give views to both these users. These users will make changes in their respective views, and the respective changes are done in the base table automatically.

2.6.1 Creating View

Consider existing table student

Table 2.6.1 : Student

roll_no	stud_name	bdate	marks
101	Kunal	12-02-2000	90
102	Jay	07-08-1999	68
103	Radhika	05-04-2000	85
104	Sagar	13-02-2000	70
105	Supriya	11-08-1999	72

1. Creating view having all records and fields from existing table

Syntax

```
CREATE or replace VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

For Example : Creating a view of base table student with same structure and all the records.

```
Create or replace view stud_view1
as select * from student;
```

Output

Table 2.6.2 : stud_view1

roll_no	stud_name	bdate	marks
101	Kunal	12-02-2000	90
102	Jay	07-08-1999	68
103	Radhika	05-04-2000	85
104	Sagar	13-02-2000	70
105	Supriya	11-08-1999	72

2. Creating view having specific fields but all the records from existing table

Syntax

```
Create or replace view view_name
as select field_1,field_2...
from existing_table_name;
```

For Example

```
Create or replace view stud_view2
as select roll_no, name from student;
```

Output : The newly created view will be

Table 2.6.3 : stud_view2

roll_no	stud_name
101	Kunal
102	Jay
103	Radhika
104	Sagar
105	Supriya

3. Creating new view having specific records but all the fields from existing table

Syntax

```
Create or replace view view_name
as select * from existing_table_name
where condition;
```

For Example

```
Create or replace view sud_view3
as select * from student
where marks > 80;
```

Output

Table 2.6.4 : stud_view3

roll_no	stud_name	bdate	marks
101	Kunal	12-02-2000	90
103	Radhika	05-04-2000	85

2.6.2 Updating View

Update query is used to update the records of view. Updation in view reflects the original table also. Means the same changes will be made in the original table also.

Syntax

```
UPDATE view_name
set field_name = new_value;
where condition;
```

For Example : We are updating marks to 73 of student having roll_no 102.

```
UPDATE stud_view1
set marks = 73
where roll_no=102;
```

In this case marks of roll_no 102 will get updated in both view view1 as well as table student

Output : View - View1

roll_no	stud_name	bdate	marks
101	Kunal	12-02-2000	90
102	Jay	07-08-1999	**73**
103	Radhika	05-04-2000	85
104	Sagar	13-02-2000	70
105	Supriya	11-08-1999	72

Output

Table 2.6.5 : Student

roll_no	stud_name	bdate	Marks
101	Kunal	12-02-2000	90
102	Jay	07-08-1999	**73**
103	Radhika	05-04-2000	85
104	Sagar	13-02-2000	70
105	Supriya	11-08-1999	72

There are some restrictions on the modification with respect to view.

- In case of view containing joins between multiple tables, only insertion and updation in the view is allowed, deletion is not allowed
- Data modification is not allowed in the view which is based on union queries.
- Data modification is not allowed in the view where GROUP BY or DISTINCT statements are used.
- In view the text and image columns can't be modified.

2.6.3 Dropping View

DROP query is used to delete a view.

Syntax

```
DROP view view_name;
```

For Example

```
DROP view stud_view2;
```

Syllabus Topic : Indexes

2.7 Indexes

SPPU - May 15

University Question

Q. Explain Index objects in SQL with example. **(May 2015, 3 Marks)**

- Sometimes the data in the database is very large. For example in the application of State Bank of India, the database related to customers and their

transactions is very large. In such case retrieval of data from such huge database becomes slower.

- Indexes are the special lookup tables which are available to only database search engine for accessing data. Indexes speed up data retrieval effectively.
- An index is a pointer to data in a table. It is similar to the alphabetical index of a book present at the end of book. An index is used to speed up SELECT queries and also WHERE clauses.
- But because of indexes the data input related to INSERT and UPDATE statements get slow down.
- Indexes can be created or dropped with no effect on the data.

Creating Index

CREATE INDEX statement is used to create an index. In this statement we have to mention name of the index, the table and column, and whether the index is in ascending or descending order.

There are different types of indexes.

2.7.1 Single Column Index

This is index is created on a single column of a table.

Syntax

```
CREATE INDEX index_name
ON table_name (column_name)
```

For Example

```
CREATE INDEX ind1
on student(stud_name);
```

2.7.2 Composite Index

Sometimes duplicate records may available in columns. In such case the composite indexing is better option to index the data. This index is created on a multiple columns of a table.

Syntax

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

For Example

```
CREATE INDEX ind2 on student(stud_name , marks);
```

2.7.3 Unique Index

A unique index does not allow any duplicate values to be inserted into the table.

Syntax

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

For Example

```
CREATE UNIQUE INDEX ind3
on student(stud_name);
```

2.7.4 Implicit Index

Implicit indexes are indexes that are automatically created by the database server when an object is created. Such indexes are created for primary key and unique constraints.

Displaying Index : To display index information regarding table following query is used.

Syntax

```
Show index from table_name;
```

For Example

```
Show index from student;
```

Syllabus Topic : SQL DML Queries - Select Query and Clauses

2.8 SQL DML Queries - Select Query and Clauses

2.8.1 SELECT Query

SELECT query is used to retrieve the data from database. SELECT query never make any change in the database. The data returned by the SELECT query is in the form of result sets.

Syntax

```
SELECT column_1, column_1... from table_name;
```

For Example : Consider the table student

Table 2.8.1 : Student

roll_no	stud_name	bdate	Marks
101	Kunal	12-02-2000	90
102	Jay	07-08-1999	68

roll_no	stud_name	bdate	Marks
103	Radhika	05-04-2000	85
104	Sagar	13-02-2000	70
105	Supriya	11-08-1999	72

```
SELECT roll_no, stud_name from student;
```

Output

roll_no	stud_name
101	Kunal
102	Jay
103	Radhika
104	Sagar
105	Supriya

The names of columns specify data from which columns we want to display. If we want data from all the columns then no need to mention column names. '*' symbol represent all the columns.

For Example

```
SELECT * from student;
```

Output

roll_no	stud_name	bdate	Marks
101	Kunal	12-02-2000	90
102	Jay	07-08-1999	68
103	Radhika	05-04-2000	85
104	Sagar	13-02-2000	70
105	Supriya	11-08-1999	72

With SELECT statement different clauses can be used to display the data as per our requirements.

2.8.2 WHERE Clause

WHERE clause is used to specify condition in SELECT statement while fetching records from the database. The WHERE clause filters the data to be retrieved. The records satisfying the condition given by where clause are retrieved.

Syntax

```
SELECT column_1,columns_2... from table_name
where condition;
```

For Example

```
select * from student where marks > 80;
```

Output

roll_no	stud_name	Bdate	marks
101	Kunal	12-02-2000	90
103	Radhika	05-04-2000	85

```
Select * from student where ename = 'Kunal';
```

Output

roll_no	stud_name	Bdate	marks
101	Kunal	12-02-2000	90

2.8.3 DISTINCT Clause

This clause is used to avoid selection of duplicate rows. Consider there are duplicate values in JOB column of emp table

Table 2.8.2 : Emp

Eno	Ename	Job	Sal
101	Susheel	Clerk	12000
102	Ajay	Manager	18000
103	Dinesh	Clerk	10000
104	Bharati	Manager	17000
105	Prajakta	Salesman	13000

Syntax

```
SELECT distinct( column_name) from table_name;
```

For Example

```
select distinct(job) from emp;
```

Output

Job
Clerk
Manager
Salesman

2.8.4 GROUP BY Clause

The GROUP BY clause is used in collaboration with the SELECT statement. It helps to arrange similar data into groups. It is also used with SQL functions to group the result from one or more tables.

Table 2.8.3 : Emp1

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

1. Display sum of salaries department wise.

```
Select deptno,sum(sal) from emp group by deptno;
```

Output

DEPTNO	SUM(SAL)
30	9400
20	10875
10	8750

2. Display maximum salaries groping on the basis of job

```
Select job,max(sal) from emp group by job;
```

Output

JOB	MAX(SAL)
CLERK	1300
SALESMAN	1600
PRESIDENT	5000
MANAGER	2975
ANALYST	3000

In general we use WHERE clause to give some condition to filter the data. But WHERE clause is not allowed in collaboration with GROUP BY clause. HAVING clause is used with GROUP by clause to specify condition.

2.8.5 HAVING Clause

1. Display sum of salaries of department 10

```
Select deptno,sum(sal) from emp group by deptno
having deptno = 10;
```

Output

DEPTNO	SUM(SAL)
10	8750

2. Display sum of salaries of department 10 and 20

```
Select deptno,sum(sal) from emp group by deptno
having deptno in (10,20);
```

Output

DEPTNO	SUM(SAL)
20	10875
10	8750

Syllabus Topic : Database Modification using SQL Insert, Update and Delete Queries

2.9 Database Modification using SQL Insert, Update and Delete Queries

2.9.1 Insert

This query comes under the category Data Definition Language. After creation of table the insert command is used to insert one or more records in the table.

Insert query has different forms
(Consider Table 2.8.2 Emp)

(1) Inserting values in all columns

Syntax

```
Insert into table_name
values (value1, value2....)
```

For example

```
Insert into emp
values(105,'Rajesh','Clerk',12000);
```

(2) Inserting values in specific columns

Sometimes we may not have all values to insert into table. In such case the syntax will be

```
Insert into table_name(column1,column2...)
values(val1,val2...);
```

For example, consider we do not have value for salary while inserting a new record in emp table. Then the query will be

```
insert into emp(Eno,Ename,Job)
values(107,'Ankur','Salesman');
```

(3) Inserting records from existing table into new table

We can also take records from existing table to add into new table using as select clause. Consider new table emp1 in which we will add records from Table 2.9.1. Here we can mention condition using where clause to take specific records.

```
Insert into emp1
Select eno,ename,job,sal from emp
Where sal > 15000;
```

Output

Table 2.9.1 : Emp1

Eno	Ename	Job	Sal
102	Dinesh	Manager	18000
104	Bharati	Manager	17000

2.9.2 Update

Sometimes changes to the database become necessary. To make changes in the database 'update' command is used. Updations can be done in single or multiple columns based on the given condition. The update command consists of 'set' clause and an optional 'where' clause'

Syntax

```
Update table_name set column_name = new_value
[where condition]
```

For Example

```
Update emp set sal = 15000;
```

This query will make salary of all the employees to 15000.

Output

Eno	Ename	Job	Sal
101	Susheel	Clerk	15000
102	Dinesh	Manager	15000
103	Dinesh	Clerk	15000
104	Bharati	Manager	15000
105	Prajakta	Salesman	15000

'WHERE' clause is used to make changes in specific records.

For Example

```
Update emp set sal = 20000 where job = 'Manager';
```

This query will change salary of mangers to 20000.

Output

Eno	Ename	Job	Sal
101	Susheel	Clerk	15000
102	Dinesh	Manager	20000
103	Dinesh	Clerk	15000
104	Bharati	Manager	20000
105	Prajakta	Salesman	15000

2.9.3 Delete

As per requirement, the records from existing table can be removed using delete command. Delete command can have 'WHERE' clause optionally.

Syntax

```
Delete from table_name;
```

For Example

```
Delete from emp;
```

This query will delete all the records from Table 2.9.1.

```
Delete from emp where Eno = 103;
```

This query will delete the record of employee with employee number 103 from emp table.

Syllabus Topic : Set Operations

2.10 Set Operations

- Set operations are supported by SQL to be performed on table data. For these operations special operators knows as Set Operators are used. Set operators are used to join the results of multiple SELECT statements.
- These operators help to get meaningful results from data, under different specific conditions. Queries which contain set operators are called compound queries.
- The different Set Operators are as follows

(i) Union (ii) Union All
(iii) Intersect (iv) Minus

Consider following two tables Emp and Dept

Table 2.10.1 : Emp

Empno	Ename	Job	DeptNo	Salary
101	Rahul	Manager	10	17000
102	Vinay	Clerk	20	12000
103	Kunal	Manager	30	18000
104	Rajesh	Salesman	20	13000
105	Kushal	Clerk	10	11000

Table 2.10.2 : Dept

DeptNo	DeptName	Loc
10	Sales	Mumbai
20	Production	Pune
30	Accounts	Nasik
40	Research	Bangalore

2.10.1 Union

The union operator returns all distinct rows selected by either query

Syntax

```
Select column_name from table_1
Union
Select column_name from table_2
```

For Example

```
Select DeptNo from emp
union
select Deptno from dept
```

Output

10
20
30
40

2.10.2 Union All

The Union All operator returns all rows selected by either query including duplicates.

Syntax

```
Select column_name from table_1
Union all
Select column_name from table_2
```

For Example

```
Select DeptNo from emp
Union all
select Deptno from dept
```

Output

10
20
30
20
10
10
20
30
40

2.10.3 Intersect

The intersect operator returns only those rows which are common to both the queries

Syntax

```
Select column_name from table_1
intersect
Select column_name from table_2
```

For Example

```
Select DeptNo from emp
intersect
select Deptno from dept
```

Output

10
20
30

2.10.4 Minus

Minus operator displays the rows which are present in the first query but absent in the second query, with no duplicates and data is arranged in ascending order by default.

Syntax

```
Select column_name from table_1
minus
Select column_name from table_2
```

For Example

```
Select DeptNo from dept
intersect
select Deptno from emp
```

Output

40

Syllabus Topic : Predicates and Joins

2.11 Predicates and Joins

2.11.1 Predicates

Predicate is an expression that evaluates to TRUE, FALSE or UNKNOWN. Predicates are used in the search condition of WHERE and HAVING clauses, the join conditions of FROM clauses and other constructs where value in Boolean format is expected.

Consider the table

Table 2.11.1 : Emp

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

SQL provides various types of predicates.

2.11.1.1 Comparison Predicate

Comparison predicate is the combination of two expressions separated by a comparison operator. There are six types of comparison operators: =, >, <, >=, <=, <>. The data of NUMERIC type is compared with respect to their algebraic values. The data of CHARACTER STRING type is compared with respect to their alphabetic order.

= Equal to Predicate

```
Select * from emp
where ename = 'KING'
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10

> Greater Than Predicate

```
Select * from emp
where sal > 3000;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10

< Less Than Predicate

```
Select * from emp
where sal < 3000;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

>= Greater Than Equal To Predicate

```
Select * from emp
where sal >= 3000;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20

<= Less Than Equal To Predicate

```
Select * from emp
 where sal <= 3000;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

<> Not Equal To Predicate

```
Select * from emp
where sal <> 3000;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

Combination of Predicates can be used with AND operator

```
Select * from emp
where sal >=3000 and sal <=5000;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20

2.11.1.2 Between Predicate

Between predicate is used to specify certain range of values. The AND keyword is used in this predicate.

Syntax

```
test_expression [ NOT ] BETWEEN begin_expression AND end_expression
```

Example

```
Select * from emp
where comm between 300 and 500;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30

In "Between" predicate the NOT keyword can also be used

```
Select * from emp
where comm not between 300 and 500;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30

2.11.1.3 In Predicate

IN predicate particularly determines whether the value of expression given to test matches any value in specified the list.

Just consider that we want display records of employees from depno 10 and 20.

Then the query will be

```
Select * from emp
where deptno in(10,20)
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7934	MILLER	CLERK	7782	01/23/1982	1300		10

2.11.1.4 Like Predicate

Like operator determines whether a specific character string matches the given pattern or not. In the pattern we can use regular characters and wildcard characters. In this pattern matching, it is necessary that regular characters must exactly match the characters specified in the character string. However, for the wildcard characters arbitrary fragment matching of the character string is done. The use of wildcard characters makes the LIKE operator more flexible.

Example : Display records of employee whose names starts with letter 'J'

Query

```
select * from emp
where ename like 'J%';
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7900	JAMES	CLERK	7698	12/03/1981	950		30

Example : Display records of employee whose names ends with letter 'N'

Query

```
select * from emp
where ename like '%N';
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30

Example : Display records of employee whose names contains 'L' as second character.

Query

```
select * from emp
 where ename like '_L%';
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30

Example : Display records of employee whose names contains character 'A' anywhere;

Query

```
select * from emp
where ename like '%A%';
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30

2.11.1.5 IS [NOT] NULL

When values for some attributes are not available then, NULL value is assigned. To display records having NULL value, IS NULL predicate is used.

Example : Display records of employees who never get any commission.

```
select * from emp
where comm IS NULL;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

The NOT keyword can be used to get values opposite to given condition

Example : Display records of employees who get commission.

```
select * from emp
where comm IS NOT NULL;
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30

2.11.2 Joins

SPPU - Dec. 13, May 14

University Questions

Q. What are different types of joins in SQL? Explain with suitable example. (Dec. 2013, 6 Marks)

Q. Explain different types of joins with example. (May 2014, 8 Marks)

- A JOIN is a means for combining columns from one (self-table) or more tables by using values common to each.
- There are following types of Joins.

1. INNER
2. OUTER
3. LEFT OUTER
4. RIGHT OUTER
5. FULL OUTER
6. SELF

To understand joins consider following two tables.

Table 2.11.2 : Emp

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

Table 2.11.3 : Dept

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

- In the database having large amount of data, it is not possible to store the entire data in a single table. The related data may be stored in different tables. In above tables emp and dept the data of employees is stored.
- The EMP table contains the basic information of employee like employee number, name, job(post), salary, department number etc. The DEPT table contains the information of same employees about their depart names and locations.
- If we want to display whole information means basic information with department names and locations then we have to combine these two tables in query using joins.
- For using joins, we required a common column between both the tables. In this example the EMP and DEPT tables contains a common column DEPTNO.

2.11.2.1 Inner Join (Equi Join)

The INNER JOIN is used to display records that have matching values in both tables.

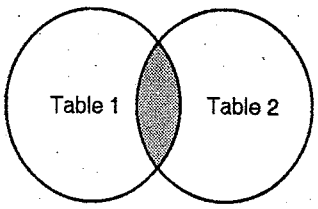


Fig. 2.11.1

Syntax

```
Select column_name_list from table_1
INNER JOIN table_2
ON table_1.column_name = table_2.column_name
```

Example

```
select ename,job,sal,emp.deptno,dept.dname from emp
INNER JOIN dept
on emp.deptno = dept.deptno;
```

Output

ENAME	JOB	SAL	DEPTNO	DNAME
CLARK	MANAGER	2450	10	ACCOUNTING
MILLER	CLERK	1300	10	ACCOUNTING
KING	PRESIDENT	5000	10	ACCOUNTING
FORD	ANALYST	3000	20	RESEARCH
SCOTT	ANALYST	3000	20	RESEARCH
JONES	MANAGER	2975	20	RESEARCH
SMITH	CLERK	800	20	RESEARCH
ADAMS	CLERK	1100	20	RESEARCH
WARD	SALESMAN	1250	30	SALES
MARTIN	SALESMAN	1250	30	SALES
TURNER	SALESMAN	1500	30	SALES
JAMES	CLERK	950	30	SALES
ALLEN	SALESMAN	1600	30	SALES
BLAKE	MANAGER	2850	30	SALES

The INNER JOIN keyword selects all rows from both tables Emp and Dept as long as there is a match between the columns. If there are records in the "Dept" table that do not have matches in "Emp", then such records will not get displayed. In Dept table department OPERATIONS of number 40 is present, but it is not displayed as no matching record is available in table emp.

2.11.2.2 Outer Join

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

(1) Left Outer Join (2) Right Outer Join (3) Full Outer Join

Consider following two tables Stud_data1 and Stud_data2

Table 2.11.4 : Stud_data1

Roll No	NAME
1	Rahul
2	Kunal
3	Jay
4	Vinay
5	Preeti

Table 2.11.5 : Stud_data2

Roll No	Address
1	Mumbai
2	Pune
3	Nasik
7	Bangalore
8	Goa

(1) Left Outer Join

The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. Null values are shown at the place of right table values.

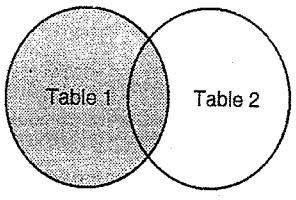


Fig. 2.11.2

Left Outer Join syntax is,

```
SELECT column-name-list
from table-name1
```

```
LEFT OUTER JOIN
table-name2
on table-1.column-name = table-2.column-name;
```

Example

```
SELECT * FROM Stud_data1
LEFT OUTER JOIN Stud_data2 ON
Stud_Data1.rollno = Stud_Data2.Rollno);
```

Output

ID	Name	ID	Address
1	Rahul	1	Mumbai
2	Kunal	2	Pune
3	Jay	3	Nasik
4	Vinay	NULL	NULL
5	Preeti	NULL	NULL

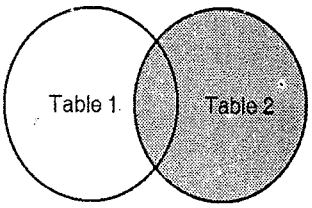


Fig. 2.11.3

(2) Right Outer Join

Returns all rows from the right table even if there are no matches in the left table. Null values are shown at the place of left table values.

Syntax

```
select column-name-list from table-name1
RIGHT OUTER JOIN table-name2
on table-1.column-name = table-2.column-name;
```

Example

```
SELECT * FROM Stud_Data1
RIGHT OUTER JOIN Stud_Data2
on (Stud_Data1.rollno= Stud_Data2. rollno);
```

Output

ID	Name	ID	Address
1	Rahul	1	Mumbai
2	Kunal	2	Pune
3	Jay	3	Nasik
NULL	NULL	7	Bangalore
NULL	NULL	8	Goa

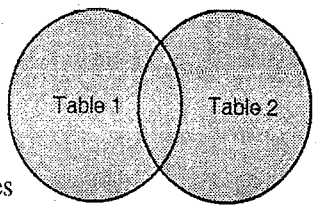


Fig. 2.11.4

(3) Full Outer Join

The full outer join returns a result with the **matching data** of both the tables and then remaining rows of both **left** table and then the **right** table.

Syntax

```
select column-name-list from table-name1
FULL OUTER JOIN table-name2
on table-1.column-name = table-2.column-name;
```

Example

```
SELECT * FROM Stud_Data1
FULL OUTER JOIN Stud_Data2
on (Stud_Data1.rollno= Stud_Data2. rollno);
```

Output

ID	Name	ID	Address
1	Rahul	1	Mumbai
2	Kunal	2	Pune
3	Jay	3	Nasik
4	Vinay	NULL	NULL
5	Preeti	NULL	NULL
NULL	NULL	7	Bangalore
NULL	NULL	8	Goa

2.11.2.3 SELF Join

- Self join is used to join a table to it-self as if the table were two tables. Virtual copies of the table are considered. Consider the table Emp

Table 2.11.6 : Emp

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

- Here the MGR number indicates the name of MANAGER of particular employee. For Example MGR of MILLER is 7782 which is EMPNO of CLARK. That mean CLARK is manager of MILLER.
- Now we want to display list of employees with their manager names. In this case we have to join this emp table to itself. We will consider two copies of emp table, emp A and emp B. From emp A we will retrieve employee names while from emp B we will get manager names;

Query

```
select A.ename "Employee" ,B.ename "Manager" from emp A, emp B where A.mgr = B.empno;
```

Output

Employee	Manager
FORD	JONES
SCOTT	JONES
ALLEN	BLAKE
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
WARD	BLAKE
MILLER	CLARK
ADAMS	SCOTT
BLAKE	KING
CLARK	KING
JONES	KING
SMITH	FORD

Ex. 2.11.1 **SPPU - Dec. 2014, 3 Marks**

Consider Following Relational Tables :
Instructor (ID, name, dept name)
Student (ID, dept_name,tot_cred)
Takes (ID, course-id, sec-id, semester, year)
Course (course –id, title, dept-name, credits)
Dept (Dept-id, dept -name)
Solve following queries using SQL
Design above relation using SQL DDL statement, primary key and foreign key.

Soln. :

```
Create table instructor(id varchar(5), name varchar(10), dept_name varchar(10), primary key(id));
Create table student(id varchar(5),dept_name varchar(10), tot_cred int(5),foreign key(id) references
instructor(id), foreign key(dept_name) references instructor(dept_name));

Create table takes(id varchar(5),course_id varchar(10), sec_id varchar(10), sem int(2), year int(2) , foreign
key(id) references instructor(id));

Create table course(course_id varchar(5),title varchar(10), dept_name varchar(10), credits int(4), foreign
key(course_id) references takes(course_id));
Create table dept(dept_id varchar(5),dept_name varchar(10),foreign key(dept_name) references
instructor(dept_name));
```

Ex. 2.11.2 **SPPU - Oct. 2016(In sem), 5 Marks**

Consider the relational database
Supplier (Sid, Sname, address)
Parts (Pid, Pname, color)
Catalog (sid, pid, cost)

Write SQL queries for the following requirements:

i) Find name of all parts whose color is green.

ii) Find names of suppliers who supply some red parts.

iii) Find names of all parts whose cost is more than Rs.25.

Soln. : Consider the following 3 tables

```
mysql> select *from supplier;
+-----+-------+---------+
| sid | sname | address |
+-----+-------+---------+
| s1  | abc   | Pune    |
| s2  | pqr   | Mumbai  |
| s3  | xyz   | Nasik   |
+-----+-------+---------+
3 rows in set (0.08 sec)

mysql> select *from parts;
+-----+-------+-------+
| pid | pname | color |
+-----+-------+-------+
| p1  | prod1 | red   |
| p2  | prod2 | green |
| p3  | prod3 | blue  |
+-----+-------+-------+
3 rows in set (0.05 sec)
```

```
mysql> select *from catalog;
+------+------+------+
| sid  | pid  | cost |
+------+------+------+
| s1   | p1   |   30 |
| s2   | p3   |   10 |
| s3   | p2   |   40 |
+------+------+------+
3 rows in set (0.00 sec)

mysql>
```

i) Find name of all parts whose color is green.

```
Select * from parts where color = green;
```

Output

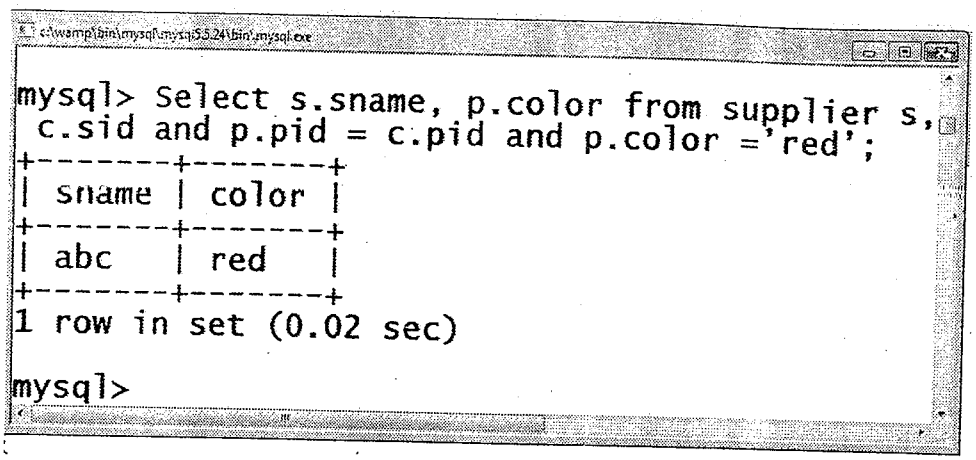
```
mysql> select* from  parts where color='green';
+-----+-------+-------+
| pid | pname | color |
+-----+-------+-------+
| p2  | prod2 | green |
+-----+-------+-------+
1 row in set (0.02 sec)

mysql>
```

ii) Find names of suppliers who supply some red parts.

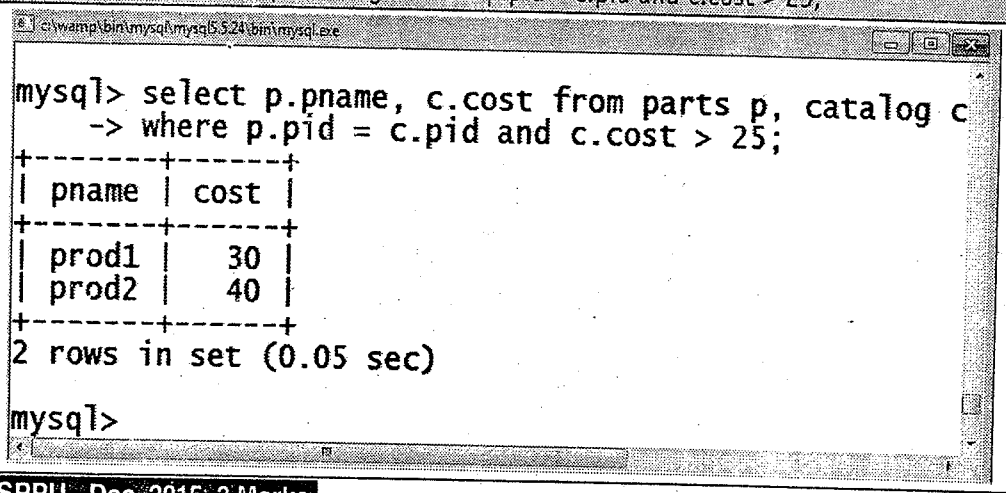
```
Select s.sname, p.color from supplier s, parts p, catalog c where s.sid = c.sid and p.pid = c.pid and p.color ='red';
```

Output



iii) Find names of all parts whose cost is more than Rs.25

```
select p.pname, c.cost from parts p, catalog c where p.pid = c.pid and c.cost > 25;
```



Ex. 2.11.3 **SPPU - Dec. 2015, 3 Marks**

Consider Following Relational Tables :

Person (pname, street, city)

works_for (pname, cname, salary)

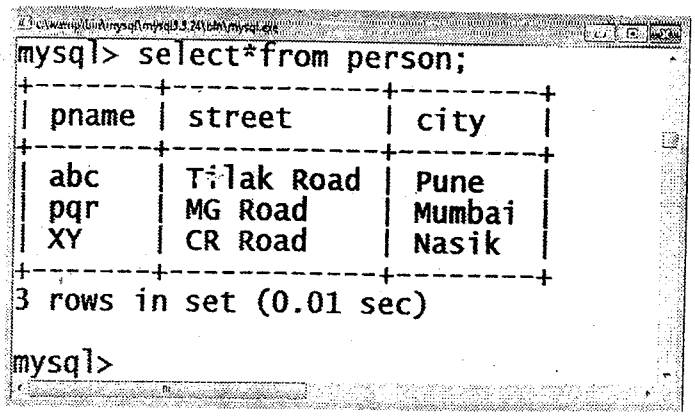
Company (cname, city)

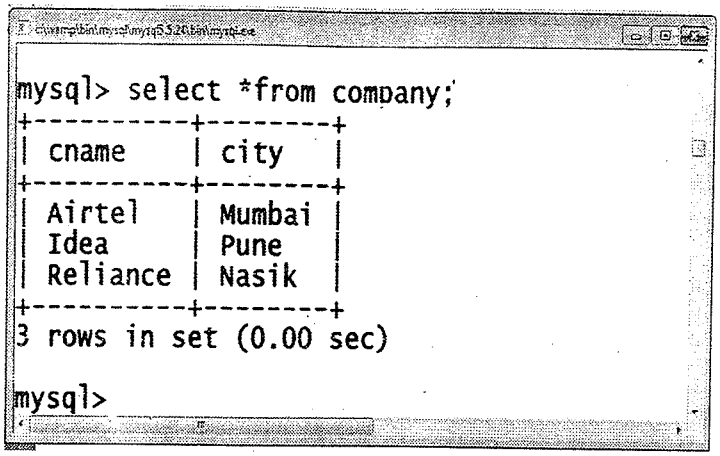
Manages (pname, mname)

Solve following quries using SQL

Find the street and city of all employees who work for the "Idea", live in Pune, and earn more than Rs. 3000.

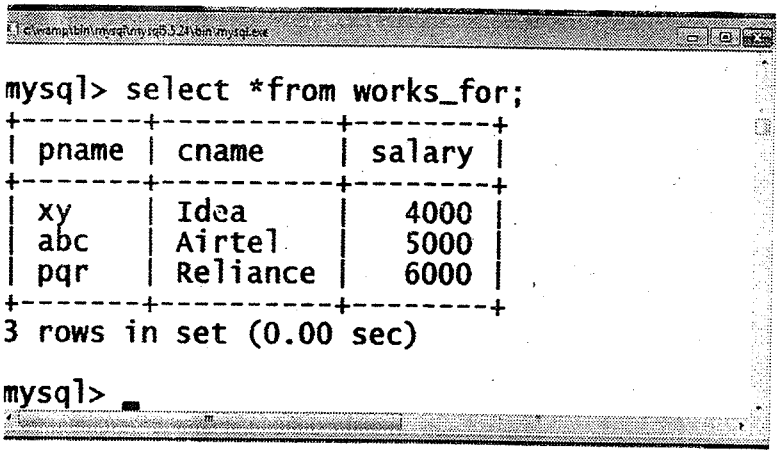
Soln. : Consider the following tables.





```
mysql> select *from company;
+----------+--------+
| cname    | city   |
+----------+--------+
| Airtel   | Mumbai |
| Idea     | Pune   |
| Reliance | Nasik  |
+----------+--------+
3 rows in set (0.00 sec)

mysql>
```



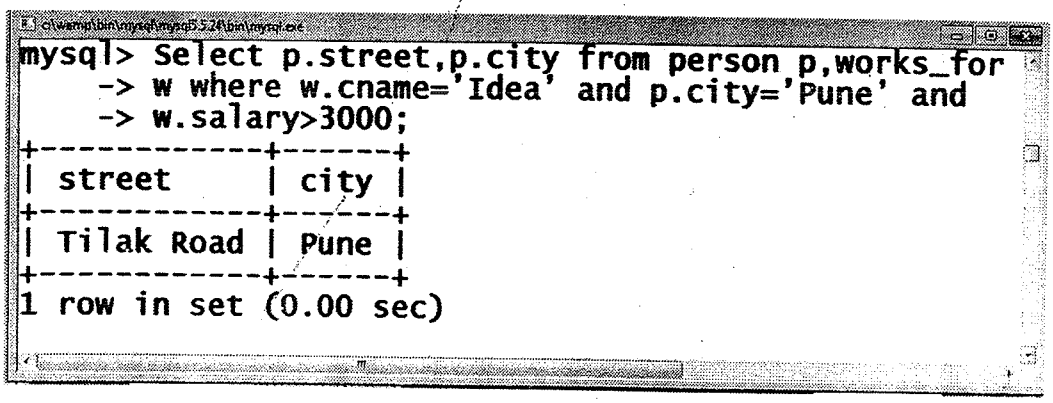
```
mysql> select *from works_for;
+-------+----------+--------+
| pname | cname    | salary |
+-------+----------+--------+
| xy    | Idea     |   4000 |
| abc   | Airtel   |   5000 |
| pqr   | Reliance |   6000 |
+-------+----------+--------+
3 rows in set (0.00 sec)

mysql>
```

i) Find the street and city of all employees who work for the "Idea", live in Pune, and earn more than Rs. 3000.

```
Select p.street,p.city from person p,works_for w where w.cname='Idea' and p.city='Pune' and w.salary>3000;
```

Output



```
mysql> Select p.street,p.city from person p,works_for
    -> w where w.cname='Idea' and p.city='Pune' and
    -> w.salary>3000;
+------------+------+
| street     | city |
+------------+------+
| Tilak Road | Pune |
+------------+------+
1 row in set (0.00 sec)
```

Ex. 2.11.4 SPPU - Dec. 2013, 4 Marks

Consider Following Relational Tables:

Student (Roll-no, name, address)

Subject (Sub_code,Sub-name)

Marks (Roll-no, Sub-code, marks)

Solve following queries using SQL

(i) Find out average marks of each student, along with the name of student.

(ii) Find how many students have failed in the subject "DBMS".

Soln. :

```
Create table student(roll_no int(3) , name varchar(10),address varchar(20), primary key(roll_no));
Create table subject(sub_code varchar(10),sub_name varchar(10), primary key(sub_code));
```

```
Create table marks(roll_no int(3) , sub_code varchar(10),marks int(3), foreign key(roll_no) references
student(roll_no), foreign key(sub_code) references subject(sub_code));
```

Consider following tables

```
mysql> select *from manages;
+-------+-------+
| pname | mname |
+-------+-------+
| xy    | aaa   |
| pqr   | bbb   |
| abc   | ccc   |
+-------+-------+
3 rows in set (0.00 sec)

mysql>
```

```
mysql> select *from subject;
+----------+----------+
| sub_code | sub_name |
+----------+----------+
| c        | C Prog   |
| jv       | Java     |
| or       | Oracle   |
+----------+----------+
3 rows in set (0.00 sec)
```

```
mysql> select *from marks;
+---------+----------+-------+
| roll_no | sub_code | marks |
+---------+----------+-------+
|       1 | or       |    90 |
|       2 | c        |    80 |
|       3 | jv       |    78 |
+---------+----------+-------+
3 rows in set (0.00 sec)
```

(i) Find out average marks of each student, along with the name of student.

```
Select st.name,su.sub_code, m.marks from student st,subject su,marks m where st.roll_no = m.roll_no and
su.sub_code=m. sub_code;
```

Output

```
+------+----------+-------+
| name | sub_code | marks |
+------+----------+-------+
| abc  | or       |    90 |
| pqr  | c        |    80 |
| xyz  | jv       |    78 |
+------+----------+-------+
3 rows in set (0.02 sec)

mysql>
```

(ii) Find how many students have failed in the subject "Java".

```
Select st.name, su.sub_name, m.marks from student st,subject su,marks m where st.roll_no = m.roll_no and
su.sub_code=m. sub_code and  m.marks > 40 and su.sub_name='Java';
```

Output : Empty Set as no student is there who failed in Java.

Syllabus Topic : Tuple variables

2.12 Tuple Variables

A Relation R can be listed number of times as per the requirements. In this situation we need a way to refer to each occurrence of R. SQL allows us to define, for each occurrence of R in the FROM clause with the help of an "alias". This alias is known as **tuple variable.** When the R is used in the FROM clause, it is followed by the keyword AS which is optional and the name of the tuple variable.

We will see the example which we have already studied in SELF JOIN

Consider the table emp

Table 2.12.1 : Emp

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

Here the MGR number indicates the name of MANAGER of particular employee.

For Example MGR of MILLER is 7782 which is EMPNO of CLARK. That mean CLARK is manager of MILLER.

Now we want to display list of employees with their manager names. In this case we have to join this emp table to itself.

We will consider two copies of emp table, emp A and emp B. From emp A we will retrieve employee names while from emp B we will get manager names;

Query

```
select A.ename,B.ename from emp A, emp B where A.mgr = B.empno;
```

Output

Employee	Manager
FORD	JONES
SCOTT	JONES
ALLEN	BLAKE
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
WARD	BLAKE
MILLER	CLARK
ADAMS	SCOTT
BLAKE	KING
CLARK	KING
JONES	KING
SMITH	FORD

Here A and B are **tuple variables.**

Syllabus Topic : Ordering of Tuples

2.13 Ordering of Tuples

To arrange the displayed rows in ascending or descending order on given field(column) , Order By Clause is used.

Syntax

```
Select * from table_name order by col1,col2..[desc]
```

For Example : We need to display the employee information as per their names in ascending order, the query will be

```
Select * from emp order by Ename;
```

Output

Eno	Ename	Job	Sal
102	Ajay	Manager	18000
104	Bharati	Manager	17000
103	Dinesh	Clerk	10000
105	Prajakta	Salesman	13000
101	Susheel	Clerk	12000

Now to display same information in descending order on job the query will be

```
Select * from emp order by Ename desc;
```

Output

Eno	Ename	Job	Sal
101	Susheel	Clerk	12000
105	Prajakta	Salesman	13000
103	Dinesh	Clerk	10000
104	Bharati	Manager	17000
102	Ajay	Manager	18000

Sometimes same records may available in field given for sorting criteria. In such case we can give names of more than one columns for sorting purpose. If data in first column is same, in such case the data of second column can be taken into consideration for sorting. Consider following table

Eno	Ename	Job	Sal
101	Susheel	Clerk	12000
102	Dinesh	Manager	18000
103	Dinesh	Clerk	10000
104	Bharati	Manager	17000
105	Prajakta	Salesman	13000

Here names of two employees is same 'Dinesh'. Now sort the data we can mention sorting fields as ename and job.

```
Select * from emp order by ename,job;
```

Output

Eno	Ename	Job	Sal
104	Bharati	Manager	17000
103	Dinesh	Clerk	10000
102	Dinesh	Manager	18000
105	Prajakta	Salesman	13000
101	Susheel	Clerk	12000

Syllabus Topic : Aggregate Functions

2.14 Aggregate Functions

Aggregate functions perform a calculation on a set of values and return a single value. Usually these functions ignore NULL values(except for COUNT).

There are different types of aggregate functions :

```
(1) Min   (2) Max   (3) Sum
(4) Avg   (5) Count
```

Consider the table emp

(1) Min() : This function returns smallest value from specified column of the table.

Query

```
Select min(sal) from emp;
```

Output

```
800
```

(2) Max() : This function returns greatest value from specified column of the table.

Query

```
Select max(sal) from emp;
```

Output

```
5000
```

(3) Sum() : This function returns sum of all the values of specified column of the tabie.

Query

```
Select sum(sal) from emp;
```

Output

```
29025
```

(4) Avg() : This function returns average of all the values of specified column of the table.

Query

```
Select avg(sal) from emp;
```

Output

```
2073.21
```

(5) Count() : This function returns total number of values of specified column of the table.

Query

```
Select count(ename) from emp;
```

Output

```
14
```

Syllabus Topic : Nested Queries

2.15 Nested Queries

Writing a query inside another query is known as nested query or subquery. The inner query gets executed first, then the output on inner query is given as input to outer query. Consider the previous emp table

Example : To display records of employees working in SMITH's department

```
Select * from emp where deptno =
(select deptno from emp where ename = 'SMITH');
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7876	ADAMS	CLERK	7788	01/12/1983	1100		20

Example : To display records of employees whose salary is more than the salary of FORD

```
Select * from emp where sal >
(select sal from emp where ename = 'FORD');
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10

Example : To display records of employees who are senior to JONES

```
Select * from emp where hiredate <
(select hiredate from emp where ename = 'JONES');
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30

Syllabus Topic : Set Membership

2.16 Set Membership

It is used to check if value of expression is matching a set of values produced by a subquery or not. There are two keywords used for SET membership – IN and NOT IN. IN is connective test for set of membership while the NOT IN is connective test for absence of SET membership.

Example – In keyword

Display records of employees working in SMITH's department

```
Select * from emp where deptno in
(select deptno from emp where ename = 'SMITH');
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7566	JONES	MANAGER	7839	04/02/1981	2975		20

Example – Not In keyword

Display records of employees who are not working in SMITH's department

```
Select * from emp where deptno not in
(select deptno from emp where ename = 'SMITH');
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7934	MILLER	CLERK	7782	01/23/1982	1300		10
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7839	KING	PRESIDENT		11/17/1981	5000		10
7900	JAMES	CLERK	7698	12/03/1981	950		30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30

Syllabus Topic : Set Comparison

2.17 Set Comparison

In nested queries, comparison operators are used with WHERE clause to specify the condition to filter the data to be display. Following are the various comparison operators

Comparison Operator	Description
=	Equal
<>	Not Equal
>	Greater Than
<	Less Than
>=	Greater Than or Equal
<=	Less Than or Equal

Example : Consider following table emp

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

< operator in nested query

Example : Display list of employees having salary less than ADAMS.

```
select * from emp where sal <
(select sal from emp where ename = 'ADAMS');
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	12/17/1980	800		20
7900	JAMES	CLERK	7698	12/03/1981	950		30

Syllabus Topic : PL/SQL

PL/SQL stands for Procedure Language / Structure Query Language. It is the combination of SQL along with the procedural features of programming languages.

PL/SQL includes procedural language elements such as conditions and loops. It allows declaration of constants and variables, procedures and functions, types and variables of those types, and triggers. It can handle exceptions (runtime errors). Arrays are supported involving the use of PL/SQL collections. It has included features associated with object-orientation. One can create PL/SQL units such as procedures, functions, packages, types, and triggers, which are stored in the database for reuse by applications.

Syllabus Topic : Concept of Stored Procedures and Functions

2.18 Concept of Stored Procedures and Functions

2.18.1 Stored Procedures SPPU - May 13

University Question

Q. Explain Stored Procedures. **(May 2013, 4 Marks)**

Stored procedure is a group of SQL statements which can be executed repeatedly. It allow for variable declarations, flow control and other useful programming techniques. Parameters are the important part of procedures. The parameters make the stored procedure more flexible and useful.

Consider the table

```
mysql> select *from  student;
+------+-------+------------+-------+
| rno  | name  | dob        | class |
+------+-------+------------+-------+
|    2 | aa    | 1990-11-21 | NULL  |
|    3 | priya | 1989-10-30 | NULL  |
|    4 | asd   | 2000-02-02 | NULL  |
|    5 | ggg   | 0000-00-00 | NULL  |
+------+-------+------------+-------+
4 rows in set (0.00 sec)

mysql>
```

Syntax

```
DELIMITER //
CREATE procedure procedure_name
([parameter(s)])
STATEMENTS
DELIMITER //
```

IN Parameter : Accepts value when procedure get called.

Example : Create a procedure which should accept rollno as parameter and display the record.

```
DELIMITER //
CREATE PROCEDURE display(IN r integer (3))
 BEGIN
 SELECT * FROM students WHERE rno = r;
 END //
DELIMITER ;
```

Now the procedure can be called as

```
CALL display(3);
```

Output

```
+------+-------+------------+-------+
| rno  | name  | dob        | class |
+------+-------+------------+-------+
|    3 | priya | 1989-10-30 | NULL  |
+------+-------+------------+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.06 sec)
```

Out Parameter : The value of an OUT parameter can be changed inside the stored procedure. The changed value is passed back to the calling program. The initial value of OUT parameter cannot be accessed by the procedure.

Example

```
DELIMITER $$
CREATE PROCEDURE display1(IN r INT,OUT nm
VARCHAR(25))
BEGIN
select name into nm from students where rno = r;
END $$
DELIMITER ;
```

Now the procedure can be called as

```
CALL display1(3 ,@n);
```

Here n is the OUT parameter which stores the name of student having rno 3;

Then execute the command

```
Select @n;
```

Output

```
+-------+
| @n    |
+-------+
| priya |
+-------+
1 row in set (0.00 sec)

mysql>
```

2.18.2 Stored Functions SPPU - May 13

University Question

Q. Explain Stored functions. **(May 2013, 3 Marks)**

Stored Function is same as of stored procedure means it is a group of SQL statements which can be executed repeatedly. It allow for variable declarations, flow control and other useful programming techniques.

Just difference is that function can return value.

Syntax :

```
CREATE FUNCTION function_name ([parameter(s)])
   RETURNS data type
DETERMINISTIC
STATEMENTS
```

Create a function to return record of student having rollno 3.

```
DELIMITER |
CREATE FUNCTION anualsal (sal int)
 RETURNS int(7)
  DETERMINISTIC
   BEGIN
    DECLARE asal int(7);
Set asal = sal * 12;
    RETURN asal;
   END|
```

Then execute the command

Select anualsal(5000) from dual;

Output

```
mysql> select anualsal(5000) from dual;
+----------------+
| anualsal(5000) |
+----------------+
|          60000 |
+----------------+
1 row in set (0.06 sec)
```

Syllabus Topic : Cursor, Trigger

2.19 Cursor, Trigger, Assertions

2.19.1 Cursor

SPPU - May 13, Dec. 13

University Questions

Q. What is cursor? Explain various types of Cursor? **(May 2013, 8 Marks)**

Q. What is cursor? Explain explicit cursor in PL/SQL with suitable example? **(Dec. 2013, 6 Marks)**

- Cursor is used to traverse in the database to access the records one by one. For example if we want to calculate the average of marks of all the students, then we can retrieve marks of every student and add in the total_marks. Cursor is just like loop concept used to traverse to every row and manipulate data.
- An area of memory(Context) is allocated for the processing of SQL statements. The context area contains information necessary to complete the processing, including the number of rows processed by the statement, a pointer to the parsed representation of the statement.
- Cursor is a handle or pointer to the context area.
- There are two types of cursors

1. Implicit Cursor
2. Explicit cursor

2.19.1.1 Implicit Cursor

- When there is no explicit cursor, the implicit cursors are created automatically whenever an SQL statement is executed. Programmers cannot control the implicit cursors and the information in it.
- When Data Manipulation statements like insert, update or delete are executed, an implicit cursor is automatically associated with these statements. In case of insert statement the data is hold by cursor while for delete and update statements, cursor identifies the rows that would be affected.
- Consider following example in which increment of 10% is given to all the employees. Here an implicit cursor is created to identify the set of rows in the table which would be affected by the update.

```
UPDATE employee
  SET salary = salary * 0.1;
```

2.19.1.2 Explicit Cursor

An explicit cursor is the one in which the cursor name is explicitly assigned to the select statement. Processing an explicit cursor involves following three steps.

(1) **Open** : The Open statement executes the select statement. Positions the cursor at the first row.

(2) **Fetch** : The Fetch statement retrieves the current row and advances the cursor to the next row for processing.

(3) **Close** : After processing of last row the cursor is disabled with the help of Close statement.

Example : Consider the table city

```
mysql> select *from city;
+-----+-----------+----------+
| id  | city_name | state_id |
+-----+-----------+----------+
| 101 | Pune      |        1 |
| 102 | Mumbai    |        1 |
| 103 | Nasik     |        1 |
+-----+-----------+----------+
```

```
DELIMITER //
CREATE PROCEDURE cname(
 IN in_state_id INT
)
BEGIN
 DECLARE record_not_found INTEGER DEFAULT 0;
 DECLARE mycity_name VARCHAR(255)
DEFAULT "";
 DECLARE mysql_cursor CURSOR FOR
    SELECT city_name FROM city
    WHERE state_id = in_state_id;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET
record_not_found = 1;
 OPEN mysql_cursor;
  cities_loop: LOOP
  FETCH mysql_cursor INTO mycity_name;
   IF record_not_found THEN
    LEAVE cities_loop;
   END IF;
  SELECT mycity_name;
 END LOOP cities_loop;
 CLOSE mysql_cursor;
END //
DELIMITER ;
```

To call the procedure give query

```
CALL cname(1);
1 is the state id.
```

2.19.2 Trigger

SPPU - May 13

University Question

Q. Explain Triggers. **(May 2013, 4 Marks)**

A trigger is a set of actions which get executed automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a particular table.

Purpose of trigger

- To generate data automatically
- Validate input data
- Replicate data to different files to achieve data consistency
- Write to other files for audit trail purposes

Syntax

```
CREATE
[DEFINER = { user | CURRENT_USER }]
TRIGGER trigger_name
trigger_time trigger_event
ON tbl_name FOR EACH ROW
trigger_body
trigger_time: { BEFORE | AFTER }
trigger_event: { INSERT | UPDATE | DELETE }
```

Consider the table employee

```
+-----+---------+------+
| eno | ename   | sal  |
+-----+---------+------+
|   1 | Rajesh  | 9000 |
|   2 | Sudhir  | 8000 |
|   3 | Radhika | 7000 |
+-----+---------+------+
3 rows in set (0.00 sec)
```

Example

```
delimiter //
CREATE TRIGGER upd_check BEFORE UPDATE ON
employee
FOR EACH ROW
BEGIN
 IF NEW.sal < 5000 THEN
 SET NEW.sal = 5000;
 ELSEIF NEW.sal > 20000 THEN
 SET NEW.sal = 20000;
 END IF;
END;//
```

Now using update query if salary is updated to less than 5000 then it will be automatically set 5000 and if it is updated to more than 20000 then it will be automatically set 20000.

Syllabus Topic : Assertion

2.19.3 Assertion

Since SQL-92, assertions have been part of the SQL standard. An assertion is an expression used to define the constraint on database that must be always true.

Difference between Assertions and check constraints is :

Unlike check constraints they are not defined on table or column level instead of that they are defined on schema level. Assertions are not defined within create table or alter table statements like check constraint. To implement some constraint at the database level such as **cross-row constraints, multi-table check constraints,** SQL assertions can be used.

To declare Cross-row constraint triggers can be used but it is error-prone, so better way use assertions on database to serve the same purpose. If an assertion is declared on some tables, then all the constraints specified by assertion are followed by the database transaction which leads to modification on those tables.

Syntax for creating assertion is as follows :

```
CREATE ASSERTION Assertion_Name CHECK
(Condition);
```

The condition is in the form of an SQL query.

Example : Bank application has many tables in their database to store data. Schema of two of them are given below :

```
Account(acc_no, branch_name, balance);
Depositor(customer_name, acc_no);
```

Create an assertion which allows customers to have maximum two accounts in a bank application.

```
CREATE ASSERSTION Num_of_Acconts CHECK
(
(SELECT COUNT(*) FROM Account, Depositor
WHERE Account.acc_no=Deposiiter.acc_no) =< 2
);
```

2.19.4 Assertion Vs Triggers

Sr. No.	Assertion	Triggers
1.	Assertion is used to specify restrictions on database, it doesn't used to modify the data.	Triggers are used to give restrictions as well as to modify the data.
2.	All assertions can be implemented as triggers.	All triggers can't be implemented as assertions.
3.	Assertions are not linked to specific tables in the database and also not linked to specific events.	Triggers are linked to specific tables and specific events.

Syllabus Topic : Roles and Privileges

2.20 Roles and Privileges

2.20.1 Roles

- This is a group of privileges that will be assigned to users : Creating a Role
- CREATE ROLE 'admin'; You can also create more than one role at once
- CREATE ROLE 'dba', 'developer', 'readonly';

2.20.2 Privileges

- **Privileges** defines the access rights to database users on database objects (like functions, procedures, or tables). They also define rights to run a SQL statement, or PL/SQL Package.

Creating New User

- There are different ways to create users with custom permissions.

  Creating new user within the MySQL shell:

  CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';

- In this situation the newly created user has no permissions to do anything with the databases. Even if the new user try to login with the given password, he will not be able to reach the MySQL shell.

- Hence the most important initial step is to provide the user with access to the information they will need.

  GRANT ALL PRIVILEGES ON *. * TO 'newuser'@'localhost';

- The * symbol indicates all the databases and tables. Means user get rights like read, edit, execute and perform all tasks on the database.

Grant Different User Permissions

The following list shows other common possible permissions that users can get.

- ALL PRIVILEGES - allows a MySQL user all access to a designated database.
- CREATE - allows user to create new tables or databases
- DROP - allows user to them to delete tables or databases.
- DELETE - allows user to delete rows from tables.
- INSERT - allows user to insert rows into tables.
- SELECT - allows user to use the Select command to read through databases.
- UPDATE - allow user to update table rows.
- GRANT OPTION - allows user to grant privileges.
- REVOKE - removes granted privileges.

Granting privileges

```
GRANT [type of permission] ON [database name].[table name] TO '[username]'@'localhost';
REVOKE privileges
REVOKE [type of permission] ON [database name].[table name] FROM '[username]'@'localhost';
Deleting user
DROP USER 'demo'@'localhost';
```

Syllabus Topic : Embedded SQL

2.21 Embedded SQL

- The method of combining of general purpose programming languages and database language like SQL is called as **embedded SQL.**
- The SQL is good for defining database structure and defining short queries but for some application it is required to mix SQL with programming language.
- In other words, SQL defines WHAT is required but it can't define HOW to meet this requirement.
- SQL is used to express queries but all the queries can't be expressed with SQL alone, so there is a need of combining database language with general purpose programming language to express such queries.

Some concepts in Embedded SQL

- **Host language :** These are programming languages (such as C, C++, COBOL, Java, etc) in which SQL statements are embedded.
- **SQL Pre-Compiler :** A pre-compiler is used to process embedded SQL statements. It is used to translate SQL statements into DBMS library calls which can be implemented into host language.
- The following code is a simple embedded SQL program, written in C

The program accepts order number from user, retrieves the customer number, salesperson, and status of the order, and displays the data on the screen.

```
int main()
{
  EXEC SQL INCLUDE SQLCA;
  EXEC SQL BEGIN DECLARE SECTION;
    int Order_ID;
    int Cust_ID;
    char Sales_Person[10]
    char Order_Status[6]
  EXEC SQL END DECLARE SECTION;

  /* Set up error processing */
  EXEC SQL WHENEVER SQLERROR GOTO qry_error;
  EXEC SQL WHENEVER NOT FOUND GOTO
invalid_number;

  /* Prompt the user for order number */
  printf ("Enter order number: ");
  scanf_s("%d", &Order_ID);

  /* Execute the SQL query */
  EXEC SQL SELECT Cust_ID, Sales_Person,
Order_status
    FROM Orders
    WHERE Order_ID = :Order_ID
    INTO :Cust_ID, :Sales_Person, :Order_Status;
```

```
    /* Display the results */
    printf ("Customer number: %d\n", Cust_ID);
    printf ("Salesperson: %s\n", Sales_Person);
    printf ("Status: %s\n", Order_Status);
    exit();

qry_error:
    printf ("SQL error: %ld\n", sqlca->sqlcode);
    exit();

invalid_number:
    printf ("Invalid order number.\n");
    exit();
}
```

- **Host Variables :** These variables are declared in section starting with BEGIN DECLARE SECTION and ending with END DECLARE SECTION keywords. These variables are prefix by (:) in an embedded SQL statement. The precompiler distinguish between host variables and database objects(like column and tables) using (:).
- **Data Types :** Usually the data types of DBMS and a host language are different. This affects the host variable because they are used by both by host language statements and embedded SQL statements. When there is no similar type in host language that corresponds to a DBMS data type, the DBMS automatically converts the data.
- **Error Handling :** Run times errors are reported by DBMS to the application program using SQL Communications Area, or SQLCA. In the above program the statement INCLUDE SQLCA includes the SQLCA structure in program. It is used when errors are processed by the programs which are returned by the DBMS.
- **WHENEVER...GOTO statement :** It tells the pre-compiler to generate error-handling code when an error occurs.

2.21.1 Advantages of Embedded SQL

1) In embedded SQL, SQL is used to handle the database and the host language handles the variables and other input and output functions.
2) In the development cycle tasks like parsing and optimizing are done which taked high overheads. It increases efficiency of CPU resources.
3) Portability is achieved. The application be precompiled on one machine and can be moved another machine for further processing.
4) There is no need that programmer should understand the complex structure of DBMS. He has create only the embedded SQL source program code.

Syllabus Topic : Dynamic SQL

2.22 Dynamic SQL

SPPU - Dec. 13

University Question

Q. Write a short note on Dynamic SQL. **(Dec. 2013, 6 Marks)**

Although static SQL works well in many situations, in some applications the data access cannot be determined in advance.

Dynamic SQL refers to SQL code that is generated in an application or from the system tables and then executed against the database to manipulate the data at run time. The SQL code is not stored in the application but rather it is collected depending upon the input given by user.

The dynamic SQL helps to develop powerful applications which allows us to create database objects and manipulate them based on the input provided by the user.

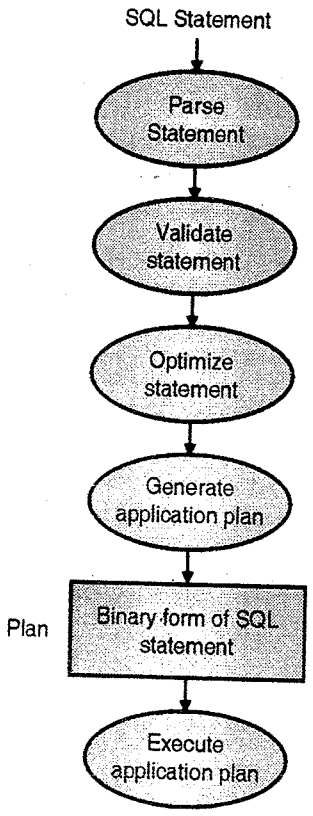


Fig. 2.22.1

In dynamic SQL the column_name or Table_name are not known at compile time so the DBMS can't able to prepare the SQL statement for execution in advance.

In this method, when the program is executed, the SQL statement get the table_name or column_name from user and this statement is given to the DBMS for further execution.

In the Fig. 2.22.1 we can observe the general execution process of dynamic sql.

An Example of Dynamic SQL statement

```
mysql> PREPARE stmt FROM
    -> 'select count(*)
    -> from information_schema.schemata
    -> where schema_name = ? or schema_name = ?'
;
Query OK, 0 rows affected (0.00 sec)
Statement prepared
mysql> EXECUTE stmt
    -> USING @schema1,@schema2
```

Output

```
+----------+
| count(*) |
+----------+
|        2 |
+----------+
1 row in set (0.00 sec)

mysql> DEALLOCATE PREPARE stmt;
```

SQL *PLUS Assignments & Solved Queries

Table 2.22.1 : Emp

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

Table 2.22.2 : Dept

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

1. Display the name and jobs of the employees who are clerk.

```
Select Ename, Job From Emp Where Job='Clerk';
```

2. Display the name and dept numbers of the employees whose job is an 'Analyst' or a 'Clerk'.

```
Select Deptno, Ename, Job From Emp Where Job In('ANALYST','CLERK');
```

3. Give the List of Employees sorted On Name.

```
Select * From Emp Order By Ename;
```

4. Display different kinds of jobs available.

```
Select Distinct Job From Emp;
```

5. Find the locations at which various Employees are situated.

```
Select Emp.Ename, Dept.Location From Emp, Dept Where Emp.Deptno= Dept.Deptno ;
```

6. Display the name, sal of the Employees whose dept locations is 'Chicago'.

```
Select Emp.Deptno, Ename, Sal From Emp Where Emp.Deptno= (Select Deptno From Dept Where
Location='Chicago') ;
```

7. Display the names of employees whose dept is 30.

```
Select Deptno, Ename From Emp Where Deptno= 30 ;
Select all Employees whose name is 5 letters long.
Select * From Emp, Where Length(Ename)=5 Order by Deptno;
Select all Employees whose name ends with R.
Select * From Emp, Where Ename like '%R';
```

8. Display the names of employees working in sales or research dept.

```
Select Ename, Deptno From Emp Where Deptno In (Select Deptno From Dept where Dname In ('SALES',
'RESEARCH'));
```

9. Display the dept no and dept name whose dept no > 20.

```
Select Deptno, Dname From Dept Where Deptno>20;
```

10. Find out dept in which maximum employees works.

```
Select Deptno, count(*) From Emp Group by Deptno Having Count(*)=(Select MAX(COUNT(*) From Emp Group
by Deptno);
```

11. Display the job, dept no, name of the employees whose name start with 'B' or 'M'.

```
Select Deptno, Ename, Job From Emp Where Ename Like 'B%' Or Ename like 'M%';
```

12. Find out the difference between maximum sal in dept 10 and minimum sal earn by a person in dept 30.

```
Select A.Sal-B.Sal From Emp A, Emp B Where A.Sal=(Select max(Sal) From Emp  where deptno=10) And
B.Sal=(Select min(Sal) From Emp  where deptno =30);
```

13. Find out the difference in dollers between the average earning of dept 20 and 30.

```
Select avg(A.Sal)-avg(B.Sal) From Emp A, Emp B group by A.Deptno,B.Deptno having avg(A.Sal)=(Select
avg(Sal) From Emp  where Deptno=20) And avg(B.Sal)=(Select avg(Sal) From Emp  where Deptno=30);
```

14. Display names, Sal and commission for Employees whose commission is more than 5% of sal.

```
Select Ename, Sal, Comm From Emp where Comm > Sal*0.05;
```

15. Find out the employees whose sal is < the average sal of dept 20.

```
Select * From Emp where Sal <(select avg(Sal) From Emp where Deptno=20);
```

16. Display information about people who is having the maximum no of people reporting to them.

```
Select * From Emp where Empno= (select MGR from Emp Group by MGR Having Count(MGR)=(select
Max(Count(MGR)) From Emp Group By MGR));
```

17. Give the query to concatenate Empno, name and manager from Emp table.

```
Select Empno||Ename||MGR From Emp;
```

18. List the employees who are hired earliest and latest.

```
Select * From Emp where HireDate=(Select Max(HireDate) From Emp) Or HireDate=(select min(HireDate) From
Emp);
```

19. List the names who are reporting to 'Blake'.

```
Select * From Emp where MGR=(Select Empno From Emp Where Ename ='Blake');
```

20. List all the employees hired in the month of December.

```
Select * From Emp where To_char(HireDate,'Month')='DEC';
```

21. List all the employees hired after 1980.

```
Select * From Emp where Tc_char(HireDate,'YYYY')>1980;
Pramote 'JAMES; to 'MANAGER' with increment of 1000
Update emp set job = manager and sal = sal + 1000 where ename = 'JONES';
```

22. Write sql command to find average annual salary per job in each department

```
Select deptno,avg(sal*12) from emp group by deptno;
```

23. Display department numbers of the departments who has more than one clerks.

```
Select deptno,count(*) from emp where job = 'CLERK' group by deptno having count(*)>1;
```

❑❑❑

UNIT III

CHAPTER 3

Relational Database Design

Syllabus

- Relational Model : Basic concepts, Attributes and Domains, Codd's Rules.
- Relational Integrity: Domain, Referential Integrities, Enterprise Constraints.
- Database Design: Features of Good Relational Designs.
- Normalization, Atomic Domains and First Normal Form, Decomposition using Functional Dependencies, Algorithms for Decomposition, 2NF, 3NF, BCNF.
- Modeling Temporal Data.

Syllabus Topic : Relational Model

3.1 Relational Model

3.1.1 Introduction

- The **Relational model** stores data in the form of tables. This concept is introduced by Dr. E.F. Codd, a researcher of IBM. The relational model is the first choice of commercial data processing applications for storing the data.
- Relational model is most famous because of its simplest structure as compare to other database models like network or hierarchical model.
- The relational model is now considered as the primary model for databases.
- The relational data model is the simple model having all the properties and capabilities required to process data.
- Most of the modern Database Management Systems (DBMS) are relational.
- The relational model consists of three major components :
  1. The set of relations and set of domains that defines the way data can be represented (data structure).
  2. Integrity rules that define the procedure to protect the data (data integrity).
  3. The operations that can be performed on data (data manipulation).
- In relational database model, the data is stored in the different tables. Theses tables are interlinked with each other with the help of common fields (columns) in between them.

History of relational model

- Edgar Codd introduced the relational model as general model of data.
- Codd proposed the relational model for IBM.
- But Codd has no idea that his influential work would become the basis of relational databases.
- The relational model is later maintained and developed by Chris Date and Hugh Darwen.
- Date and Darwen shows that the relational model contains some desired object oriented features in the third Mainfesto.

Definition of Relational Model

- A relational database is collectively combination of data structures, storage and retrieval operations and integrity constraints.
- In such a database the data and relations between them are organized into tables.
- Table is the collection of records.

3.1.2 Characteristics of Relational Database

The characteristics of Relational database systems are as follows :

- This model is called as Relational Model by Dr. Codd because the data is stored in the tables which are having relationships in between them.
- The whole data of the system is represented as systematic arrangement of data into rows and columns, called as relation or table.
- A table is also form in two-dimensional structure.
- At any given row/column position means in every cell in the relation there is one and only one value which is known as scalar value.
- Column represents attribute, and each column has a distinct name.
- All values entered in the columns are of the same data format.
- Implements the concept of closure means all operations are performed on an entire relation and result is an entire relation.
- It supports the operations like data definition, data manipulation and transaction management.

3.1.3 Advantages of Relational Model

1. **Ease of use :** The system which is managed in the form of tables consisting of rows and columns is much easier to understand.
2. **Flexibility :** The information from multiple tables can be retrieved easily at a time by joining the tables. Also changes can be done easily by using different operators.
3. **Security :** The different users can have different levels of access to data based on their roles. In the college database, students will have access to their own data only, while their teachers will have access to data of all the students to whom they are teaching. Class teacher will be able to see the reports of all the students in that class, but not other classes. The principal will have access to entire data.
4. **Data Independence :** In Relational system we can completely separate the data structure of database and programs or applications which are used to access the data. This is called as data independence. If any changes are made in structure of database then there is no need to make changes in the programs. For example you can modify the size or data type of a data items (fields of a database table) without making any change in application.
5. **Data Manipulation Language :** In the relational database approach it is easy to respond query by means of a language like SQL based on relational algebra and relational calculus. For data organized in other structure the query language either becomes complex or extremely limited in its capabilities.

Syllabus Topic : Basic Concepts, Attributes and Domains

3.1.4 Basic Concepts of Relational Model

Tables

- Relational model refers table as its basic unit.
- In relational model related data is collectively carry in the structured format in the database system.
- It is combination of rows and columns, where records are represented by rows and attributes are represented by columns. In relational data model, relations are saved in the format of Tables. The relations among entities is stored in this format.

Tuple

- Tuple is a single row of the table. It contains a whole record of a particular data item. For example in Student table the entire information about a particular student like his roll number, name, marks etc. are included in the tuple.
- A tuple is an ordered set of attribute values.

Attribute

- It is column of a table. For example in a table Student, there may be different attributes like roll_number, stud_name, marks etc.
- Every attribute is attached with specific data type which decides that which type of values can be inserted in those attributes.
- Attributes are also known as fields.

Domain

- Every attribute has some pre-defined value scope, known as attribute domain.
- This attribute domain decides which types of values are permitted in the attribute.
- The type of value contains where string or numerical or date type. The domain also decides some integrity constraints for values to be inserted

in the database. For example a emp_name attribute can have only string type of value while the salary attribute can have only numeric values with specified range.

- Domain restrict the data to be inserted with specific constraints, hence it is called as domain constraint.

Properties of relational database model

- Data is presented in the relational database model like it is the collection of relations.
- Each relation is considered as table.
- Columns are attributes that belong to the entity modelled by the table.
- Each row (Tuple) represents a single entity.
- Every table has a set of attributes collectively called as a "key" which uniquely identifies each row.

Syllabus Topic : Codd's Rules

3.2 Codd's Rules SPPU - May 14

University Question

Q. Explain in detail CODD's Rules ?
(May 2014, 8 Marks)

Codd designed these rules to define what is required from the relational database management system.

All these thirteen rules are followed by very few commercial products. Even the oracle follows eight and half rules out of thirteen.

Codd's rule are

Rule 0 : Foundation Rule

Foundation rule states that the system must be capable to manage their database systems through their relational capabilities. All other twelve rules are derived from this foundation rule.

Remaining twelve rule are as follows :

Rule 1 : Information Representation

- All the information in the database must be stored in standard form of tables.
- Table is considered as best format to store and manage the data.

Rule 2 : Systematic treatment of null values

- In a database the NULL values must be given a systematic and uniform treatment. In number of cases we may have to set null values on place of data. For Example, data is missing, data is not known, or data is not applicable.
- Null Value is different from empty character or string, string of a blank character, and it is also different from zero value or any number.

Example

EMPLOYEE			
EMP_ID	NAME	DEPARTMENT	PH_NO
E101	Naren	Testing	9656865232
E102	Atharv	NULL	9421589654
E103	Sarang	Testing	NULL

In relation PH_NO of Sarang is NULL.

Rule 3 : The guaranteed access rule

- In a data base every single data element must be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value).
- There should not be requirement of any other entity to access the data.

Rule 4 : Active online catalog

- The description of the structure of entire database must be stored in an online catalog, known as **data dictionary**.
- Data dictionary can be accessed by authorized users. Metadata should be maintained for all the data in the database.
- The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language.
- That is nothing but, just like ordinary data the database description is represented at the logical level, so that is possible for authorized users to apply the same language of regular data to its interrogation.

Rule 5 : The comprehensive data sub language rule

Database should not be directly accessible. It should always be accessible by using some strong query language. This rule illustrates that the system should support at least one relational language. The language -

a) has a linear syntax

b) can be used both interactively and within application programs,

c) 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).

Rule 6 : View updating rule

- Views are the virtual tables created using queries to show the partial or complete view of the table.
- All views that are theoretically updatable must also be updatable by the system
- The rule states that we should be able to make changes in views.

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

- High Level means multiple rows from multiple columns are affected by the single query.
- This rule states that a database must support insertion, updation, and deletion.
- This must not be limited for a single row, that is, it must also support union, minus and intersection operations to yield sets of data records.

For example,

- Suppose employees got 5% hike in a year. Then their salary has to be updated to reflect the new salary. Since this is the annual hike given to the employees, this increment is applicable for all the employees.
- Hence, the query should not be written for updating the salary one by one for thousands of employee. A single query should be strong enough to update the entire employee's salary at a time.

Rule 8 : Physical data independence

- Changes in the physical level (i.e. format of data storage or container of data like array or linked list) must not require any change to an application.

Rule 9 : Logical data independence

- The user's view (application) should not be dependent upon logical data in a database.
- That means changes in logical data must not affect the applications which uses it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application.

Rule 10 : Integrity independence

- The integrity constraints must be specified independently from application programs and stored in the catalog.
- We should be able to make changes in integrity constraints independently without the need of any change in the application.

Rule 11 : Distribution independence

- The distribution of database at various locations should not be visible to end user.
- Users should always get the impression that the data is located at one site only.
- That means even though the data is distributed, it should not affect the speed of access and performance of data compared to centralized database.

Rule 12 : The non-subversion rule

- If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language.
- Means anyhow those integrity rules and constraints must be followed, we cannot violet them by using any back door option.

Syllabus Topic : Relational Integrity

3.3 Relational Integrity

SPPU - May 13, Dec. 13

University Question

Q. How Data Integrity Problem is handled with DBMS? **(May 2013, Dec. 2013, 2 Marks)**

- Mainly security and integrity of a database is the most important factors in judging the success of system.

- Integrity constraint is a mechanism to prevent invalid data entry into table to maintain the data consistency.
- Constraints are used to enforce limits to the range of data or type of data that can be inserted/updated/deleted from a table.
- The whole purpose of constraints is to maintain the **data integrity** during the various transactions like update/delete/insert on a table.
- There are different types of constraints

1. Domain Integrity Constraints
2. Entity Integrity Constraints
3. Referential Integrity Constraints
4. Enterprise Constraints

Syllabus Topic : Relational Integrity - Domain

3.3.1 Domain Integrity Constraints

- The domain constraints are considered as the most basic form of integrity constraints. To the attributes a domain of permitted values is associated.
- For attribute it defines the default value, the range value or specific value.
- The domain integrity constraints are easy to test when data is entered.
- The domain integrity constraints check that whether the attribute having proper and right value in the database or not.
- Domain integrity means it is the collection of valid set of values for an attribute.

Constraints

1. Not Null
2. Unique
3. Default
4. Check

Data Type

- A domain is the set of all unique values which are permitted for an attribute.
- Domain constraints are user defined data type. As we say that domain is the set of unique values, the column for which Domain Constraint has set, contains same type of data, based on its data type.
- The column does not accept values of any other data type.

Not allowed as it is integer attribute.

Emp_Id	Name	Salary
8001	Rahul	15000
8002	Seema	15000
A	Devika	14000

3.3.1.1 NOT NULL

- By setting the NOT NULL constraint we can assure that a column does not hold a NULL value.
- When for a specific column, no value is provided while inserting a record into a table, by default it takes NULL value.
- NULL constraint is applied to avoid insertion of any null value in the specific column.

Example

Consider table student having 'name' field with NOT NULL constraint.

Roll_No	Name
1	Rahul
2	Seema
3	

Not allowed. Because we set name as not null constraint.

3.3.1.2 UNIQUE

- UNIQUE Constraint as the name suggests, it can take only unique values in a column or set of columns. It keeps uniqueness of the table.
- When a column has a unique constraint then that particular column cannot have duplicate values in it.
- **Example :** We can set the UNIQUE Constraint for emp_id column in employee table, each employee should have different emp_id which means this column cannot have duplicate values.

Not allowed. Because Emp_id has unique constraint

Emp_Id	Name	Salary
8001	Rahul	15000
8002	Seema	15000
8002	Devika	14000

3.3.1.3 DEFAULT

When a user does not provide a value to the column while inserting the records in the table, the DEFAULT constraint provides a default value to that column.

Example

We can set DEFAULT Constraint by assigning the value 10000 to the column exam_fees in student table. So that, if user doesn't give any value for that column, it takes the default value as 10000.

3.3.1.4 CHECK

This constraint is used to set user defined constraint for the column. As per the requirements of business for which we are developing the application, we may have to set some rules while inserting or updating data on specific field.

Roll_no	Name	Age
1	Rahul	15
2	Seema	16
3	Devika	35

Not allowed. Because range is between 15 to 20

3.3.2 Entity Integrity Constraints

3.3.2.1 Primary Key Constraint

- Under Entity Integrity Constraint Primary key is the main factor.
- Primary key uniquely identify each record in a table. It must have unique values and cannot hold null values i.e. Primary key is the combination of NOT NULL & UNIQUE constraints.

Emp_id	Name	Salary
8001	Rahul	15000
8002	Seema	15000
	Devika	14000

Not allowed as primary key not contain null.

- **Example :** Above example shows how entity integrity constraint exactly works.
- Here we set primary key to emp_id table. If we add any repeated value or null value in the column it will show error because primary key never contains null or repeated values.

Syllabus Topic : Relational Integrity - Referential Integrity

3.3.3 Referential Integrity Constraint

3.3.3.1 Foreign Key

SPPU - May 13

University Question

Q. Explain the need of Foreign Key. (May 2013, 3 Marks)

- The Foreign key constraint is also known as Referential Integrity Constraint. In this constraint one field is common in between two tables.
- Foreign key represent relationships between tables. There is parent child relationship between two tables having common column.
- The master table can be referenced as parent while the transaction table is considered as child. The common field will work as primary key in parent table while foreign key in child table.
- **Example :** Consider Training Institute Database having two tables Course_details and Student. There is a condition that the students may register for courses which are available in institute currently and not for the courses which are not offered at the moment. To specify this rule while inserting values into database foreign key constrain is used. As follows :

Course_details (Master / Parent) Table

Course_Code	Course_Name	Fees
Or	Oracle	5000
Jv	Java	4000
Cp	C Programming	3000

Student(Transaction / child) Table

Stud_ID	Name	Course_Code
S101	Kunal	Jv
S102	Radhika	Or
S013	Kiran	Cp

- In both the tables, the field **Course_Code** is common. In Course details Course_Code is referred as primary key and in Student table it is referred as foreign key.
- So, after assigning foreign key constraint to Student table the record entry for new student will not accept Course_Code which is not available in the master table Course_details.

3.3.3.2 Difference between Primary Key Constraint and Foreign Key Constraint

SPPU - May 14

University Question

Q. Differentiate between primary Key constraint & Foreign key constraint. (May 2014, 4 Marks)

Sr. No.	Primary Key	Foreign Key
1.	Primary key uniquely identify a record in the table.	Foreign key is a field in the table that is primary key in another table.
2.	Primary Key can't accept null values.	Foreign key can accept null values.
3.	By default, Primary key is clustered index and data in the database table is physically organized in the sequence of clustered index.	Foreign key do not automatically create an index, clustered or non-clustered. You can manually create an index on foreign key.
4.	We can have only one Primary key in a table.	We can have more than one foreign key in a table.

On Delete Cascade

If a record in the parent table is deleted then the corresponding records in the child table will automatically deleted. This is called as on delete cascade.

Example

- Consider Training Institute Database having two tables Course_details and Student.
- There is a condition that the students can register for courses which are available in institute currently and not for the courses which are not offered at the moment. To specify this rule foreign key constrain is used.

Table 3.3.1 : COURSE_DETAILS (Master / Parent Table)

COURSE_CODE	COURSE_NAME	FEES
Or	Oracle	5000
Jv	Java	4000
Cp	C Programming	3000

Table 3.3.2 : STUDENT(Transaction / child Table)

STUD_ID	NAME	COURSE_CODE
S101	Kunal	Jv
S102	Radhika	Or
S013	Kiran	Cp

- In both the tables, the field COURSE_CODE is common. In Course details COURSE_CODE is referred as primary key and in STUDENT table it is referred as foreign key.
- Now if we try to delete any record from master table COURSE_DETAILS, then it will show error and force us to delete all corresponding records from child table student.
- But in case of **on delete cascade** rather than showing error, all the corresponding records from child table STUDENT will get automatically deleted when record from parent table COURSE_DETAILS is deleted.

Syllabus Topic : Relational Integrity - Enterprise Constraints

3.3.4 Enterprise Constraints

- Enterprise Constraints are also referred as Semantic constraints. They are additional rules specified by users or database administrators.
- These rules are depending upon the requirements and constraints of the business for which the database system is being maintained.
- For Example, In College System a class can have a maximum of 30 students. A teacher can teach a maximum of 4 classes a semester.
- In Corporate System an employee cannot take a part in more than 5 projects. Salary of an employee cannot exceed the salary of the employee's manager etc.
- Some other examples of business rules are :
  - o A class can have a maximum of 35 students.
  - o A course can be taught many times, but by only one instructor.
  - o Not all teachers teach classes, etc.

Syllabus Topic : Database Design

3.4 Database Design

- Database design is very important aspect because properly designed database provides you with access to up-to-date, accurate information.
- Correct design is essential to achieve goals while working with a database. Investing the time required to learn the principles of good design makes sense.

- It is very important to design the database in such a manner that it should meet all our requirements and should be able to accommodate any change easily.
- Mainly a good, effective database design helps the development team to reduce the costs and time taken for the overall development.
- By creating a good data model and following the correct process, helps the development team to understand user requirements clearly and accurately.
- While designing any database we have to go step by step.
- The basic elements of the design process are :

1. Defining the problem or objective
2. Researching the current database
3. Designing the data structures
4. Constructing database relationships
5. Implementing rules and constraints
6. Creating database views and reports
7. Implementing the design

1. Defining the problem or objective

This is the first and important step of database design in which we will address the problem or objective of the database. Here the nature of data to be stored is decided.

2. Researching the current database

In some cases, there may be some database already in exist. Such database may be in any format like written on papers, spreadsheets, word files etc. The existing database is always useful for the end user. This existing database helps to determine the essential data structure of the database.

3. Designing the data structures

A database system is always a set of data tables, hence the next step in the design process is to identify and describe those data structures. The tables in the database represents some distinct subject or any physical object. By determining the subjects and objects in the system we can create list of tables to design. Then attributes of each table are decided.

4. Constructing database relationships

After creation of data structures, we have to establish relationships between the databases. It is necessary that each table should have a unique key which can identify the individual records in that table.

5. Implementing rules and constraints

Now as per the database and business requirements, some rules are set to restrict the data of insertion and modification so as to maintain data consistency. These rules refine the data and avoid any invalid data insertion or modification.

6. Creating database views and reports

Up till now the database design has completed, so we have to create views and reports to convert the data into useful information. This is usually for the end user. It helps to provide a simple structure of only required data to user by hiding the complexity and unnecessary data.

7. Implementing the design in software

This is the last and final step in which the create design is implemented through the software on the computer.

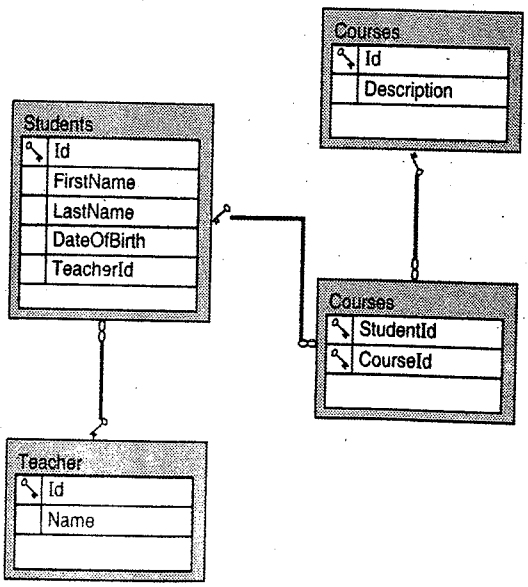


Fig. 3.4.1

Fig. 3.4.1 is simple example for database designing. Fig. 3.4.1 contains four tables students, courses, teachers, StudentCourses. It also shows the relationship between them. Here key sign shows the primary key.

Syllabus Topic : Features of Good Relational Design

3.4.1 Features of Good Relational Designs

- It should be able to distribute the information into different tables to reduce redundancy of data.
- It should provide easy access to the data available in multiple tables.
- It should take care of accuracy and integrity of information.
- It should provide functionalities for data processing and reporting needs.
- Data entry, updates and deletions should be efficient.
- Data retrieval, summarization and reporting should also be efficient.
- The database design must be self-documenting since much of the information is stored in the database rather than in the application.
- To understand the features of a good relational design we will see an example.

Example

- The relational design of Banking
  - branch : (brnch_name, brnch_city, assets) customer : (cust_id, cust_name, cust_street, cust_city)
  - loan : (loan_num, amt)
  - account : (acc_num, balance)
  - employee : (emp_id. emp_name, telephone_num, start_date)
  - dependent_name : (emp_id, dname)
  - account_branch : (acc_num, brnch_name)
  - loan_branch : (loan_num, brnch_name)
  - borrower : (cust_id, loan_num)
  - depositor : (cust_id, acc_num)
  - cust_banker : (cust_id, emp_id, type)
  - works_for : (worker_emp_id, manager_emp_id)
  - payment : (loan_num, payment_num, payment_date, payment_amt)
  - savings_account : (acc_num, interest_rate)
  - checking_account : (acc_num, overdraft_amount)
- Consider we want to get information from borrower and loan, then after combining these two relations -

  bor_loan = (cust_id, loan_num, amt)
- In the result, repetition of information is possible.
- Result is possible repetition of information. Observe **Ln101** in Fig. 3.4.2.

(A) Combine Schemas

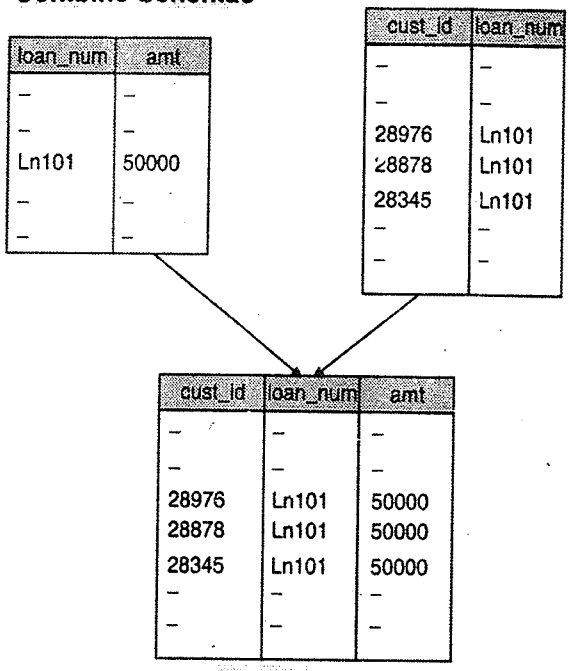


Fig. 3.4.2

1. Combining schema without repetition

Consider we want to get information from loan_branch and loan, then after combining these two relations -

loan_amt_br = (loan_num, amt, brnch_name)

Here no repetition of data will occur.

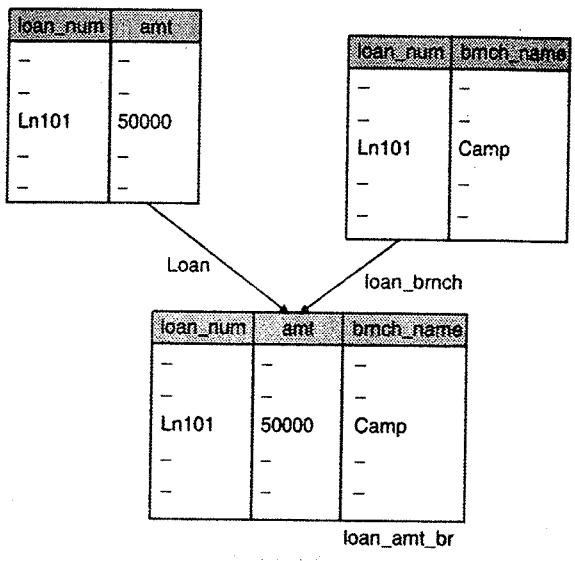


Fig. 3.4.3

B) Smaller Schemas

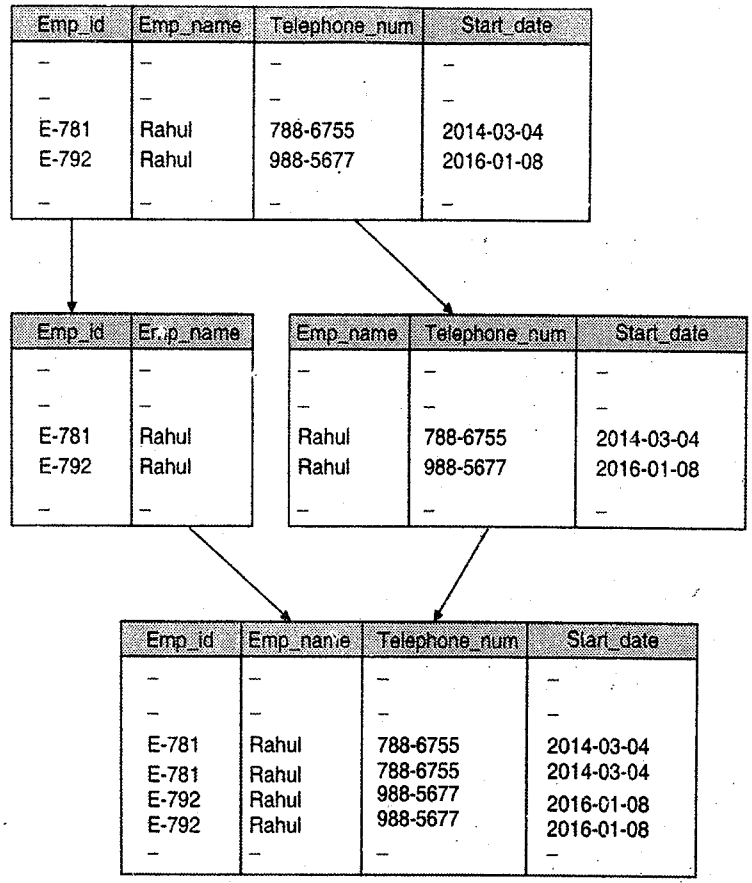


Fig. 3.4.4

- Consider that we have created bor_loan first. Now we want to split (decompose) it in borrower and loan.
- Decide a rule for a schema (loan_num, amt), loan_num would be a candidate key.
- Denote as a functional dependency : amount→loan_number
- In bor_loan relation the loan_num is not a candidate key hence the amount of a loan may have to be repeated. This indicates that we have to decompose bor_loan.
- It is not necessary that always all decompositions are good.
- Suppose we decompose employee into

employee1 = (emp_id, emp_name)

employee2 = (emp_name, telephone_num, start_date)

See the Fig. 3.4.4 which indicates how we lose information. We cannot reconstruct the original employee relation. This is called as lossy decomposition.

Syllabus Topic : Normalization

3.5 Normalization

SPPU - Dec. 14, Dec. 15, May 16

University Questions

Q. Define Normalization. Explain three normal forms with suitable example? **(Dec. 2014, 5 Marks)**

Q. Define Normalization. **(Dec. 2015, 2 Marks)**

Q. Define Database normalization. Explain any two normal forms with suitable example ? **(May 2016, 5 Marks)**

- Dr. Edgar F. Codd proposes normalization as an integral part of a relational model. **Normalization** is a database design technique which is used to organize the tables in such manner that it should reduce redundancy and dependency of data.
- **Normalization** is a database design technique which is used to organize the tables in such manner that it should reduce redundancy and dependency of data.

- It divides larger tables to smaller tables and links these smaller tables using their relationships.
- Normalization is implemented by following some formal rules either by a process of synthesis or decomposition. Synthesis is used to create database design in normalized form based on a known set of dependencies. Decomposition generally done on existing database which is not normalized. The relation of this database is further divided into multiple relations to improve the design.
- Normalization is the multi step process. It creates smaller tables from larger table.

Types of Normalization

1) **First normal form (1NF) :** Having unique values, no repeating groups.
2) **Second Normal form (2NF) :** Having unique values, no repeating groups, no partial dependency.
3) **Third Normal form (3NF) :** Same like second normal form and having transitive dependency.
4) **Boyce-Codd Normal form (BCNF) :** It is more developed version then 3NF.
5) **Fourth normal form (4NF) :** No multi-valued dependency.

3.5.1 Need of Normalization

- In database management process without Normalization, it is not easy to manage database operations like insertion, deletion, and modification without facing data loss problem.
- If database is not normalized, then Insertion, Updation and Deletion Anomalies occur frequently. To understand need of normalization, first we should learn, what are **Anomalies** ?

3.5.1.1 Anomalies

Anomalies are inconvenient or error-prone situation arising when we process the tables. There are three types of anomalies :

A) Insert Anomaly

- An **Insert Anomaly** occurs when it is not possible to insert certain attributes into the database without the availability of other attributes.
- For example, In College Database System, it is not possible to add entry of any new course unless any student enrolled for it.

Table 3.5.1 : STUDENT_INFO

STUD_ID	COURSE_ID	STU_NAME	ADDRESS	COURSE
S101	C1	Kunal	Camp	VB
S101	C2	Kunal	Camp	Java
S102	C3	Radhika	Sahkar Nagar	Oracle
S103	C4	Kiran	Tilak Road	C++
S104	C5	Jay	Pune Station	DS
S105	C3	Pooja	Sadashiv Peth	Oracle

- Here no student has enrolled for the course Dot Net, hence no entry for course Dot Net.

B) Update Anomaly

- An **Update Anomaly** occurs when there is requirement of changes in multiple records of an entity, but all records not get updated.
- For Example : Address of Kunal get changed.

Table 3.5.2 : STUDENT_INFO

STUD_ID	COURSE_ID	STU_NAME	ADDRESS	COURSE
S101	C1	Kunal	Tilak Road	VB
S101	C2	Kunal	Camp	Java
S102	C3	Radhika	Sahkar Nagar	Oracle
S103	C4	Kiran	Tilak Road	C++
S104	C5	Jay	Pune Station	DS
S105	C3	Pooja	Sadashiv Peth	Oracle

C) Delete Anomaly

- A **Delete Anomaly** is opposite to insert anomaly. This anomaly occurs when data of some attributes get lost because of deletion of data of other attributes in the same record.
- For Example : Just consider the only one entry for course Oracle of student Jay is deleted, then there will be no record related to course 'DS'.

STUD_ID	COURSE_ID	STU_NAME	ADDRESS	COURSE
S101	C1	Kunal	Tilak Road	VB
S101	C2	Kunal	Camp	Java
S102	C3	Radhika	Sahkar Nagar	Oracle
S103	C4	Kiran	Tilak Road	C++
S105	C3	Pooja	Sadashiv Peth	Oracle

- To overcome these anomalies we need to normalize the data in database.

Syllabus Topic : Atomic Domains and First Normal Form

3.6 First Normal Form (1NF)

- A **domain** is the set of all unique values which is permitted for an attribute. **Atomic** means that cannot be divided further.
- First Normal Form defines that all the attributes in a relation must have atomic (not further divisible) and single values.
- Consider the table Course _details

Table 3.6.1 : Course_details

Category_Id	Category_name	Languages
C_PRG	Programming	C, VB, Java
C_SCR	Scripting	JavaScript, PHP, HTML

- Table 3.6.1 is **not in 1NF** as the rule says "each attribute of a table must have atomic (single) values", the attribute 'Languages' contain multiple values which violets the rule of 1NF. To convert this data into First Normal Form, we have to rearrange it in the table.

Category_Id	Category_name	Languages
C_PRG	Programming	C
C_PRG	Programming	VB
C_PRG	Programming	Java
C_SCR	Scripting	JavaScript
C_SCR	Scripting	PHP
C_SCR	Scripting	HTML

- Now each attribute contain single values. Hence the database design is in First Normal Form.

Syllabus Topic : Decomposition using Functional Dependencies

3.7 Decomposition using Functional Dependency

3.7.1 Functional Dependency

The attributes of a relation is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table. This is called as **functional dependency**.

If attribute A of a relation uniquely identifies the attribute B of same relation then it can represented as $A \rightarrow B$ which means attribute B is functionally dependent on attribute A.

Formal definition of functional dependency

Let, R be a relation schema with attributes A, B i.e. R include A, B.

Given a relation R,

A set of attributes B in R is said to functionally dependent on another set of attributes A in R,

i.e. $A \rightarrow B$

Left-hand side of FD that is A is the determinant set of attributes and right-hand side of FD means B is the dependent or determined set of attributes. Usually primary key of relation is determinant in FD. Thus, given a row and the values of the attributes in set A, one can determine the corresponding value of the B attribute.

Table 3.7.1 : STUDENT

STUDENT_ID	STUDENT_NAME	ADDRESS	COURSE
1001	Akash	Nagpur	Oracle
1002	Anuja	Pune	Java
1003	Smita	Jalgaon	Php

From STUDENT_ID attribute it is possible to identify particular students' name. Name of the student is functionally dependent on STUDENT_ID and not vice versa; because more than two students may have same names and using STUDENT_ID we can determine the name of student but using STUDENT_NAME we can't determine the appropriate STUDENT_ID.

So we can say that,

$$STUDENT\_ID \rightarrow STUDENT\_NAME$$

i.e. STUDENT_NAME is functionally dependent on STUDENT_ID

Similarly,

STUDENT_ID → all the attributes(i.e. ADDRESS, COURSE)

So we can say that all the remaining attributes are functionally dependent on STUDENT_ID attribute.

It is possible that an attribute of a relation can be identified by set of another attributes which are of the same relation. This can be implies as follows:

STUDENT_ID, STUDENT_NAME → ADDRESS

STUDENT_ID, STUDENT_NAME → COURSE

In above situation, usually composite primary key of relation is determinant in FD.

3.7.1.1 Types of Functional Dependencies

A) Single-valued Functional Dependency
B) Multi-valued Functional Dependency
C) Fully Functional Dependency
D) Partial Functional Dependency
E) Transitive Functional Dependency
F) Trivial Functional Dependency
G) Non Trivial Functional Dependency

A) Single-valued Functional Dependency

- Database is a collection of related information in which information depends on another information.
- The information is either single-valued or multi-valued. For example, the name of the person or his / her date of birth is single valued facts. But the Contact number of a person is a multi-valued attribute.
- A single valued functional dependency is when A is the primary key of an relation (e.g. STUDENT_ID) and B is some single valued attribute of the relation (e.g. STUDENT_NAME). Then, A → B must always hold by the relation.

STUDENT

STUDENT_ID	STUDENT_NAME	ADDRESS	DOB	COURSE
1001	Akash	Nagpur	11-2-1980	Oracle
1002	Anuja	Pune	2-10-1982	Java
1003	Anuja	Jalgaon	23-10-1981	PHP
1004	Amruta	Nanded	20-9-1981	DBMS

STUDENT_ID → STUDENT_NAME

1002 Anuja

For every STUDENT ID there should be unique name (A → B)

B) Multi-valued Functional Dependency

SPPU - May 14

University Question

Q. Write short note on : Multivalued Dependency. **(May 2014, 4 Marks)**

- A **multi-valued dependency** exists when a relation R has at least 3 attributes (like A, B and C) and for value of A there is a well defined set of values of B and a well defined set of values of C. However, the set of values of B is independent of set C and vice versa.

- For Example :

Table 3.7.2 : Published_Book

Book_Id	Auther_Name	Price
B_001	Jasmin	1000
B_002	Jasmin	980
B_003	Smita	1500
B_004	Smita	2000

- As shown in Table 3.7.2, name of particular author is determined by Book_Id attribute.

  {Book_Id} ->>{Author_Name}

- Also it is possible to determine the Price of book by Book_Id.

  {Book_Id} ->> {Price}

- But it is difficult to determine the price of book using author name or vice versa. So this dependency is called as multi-valued functional dependency or simply multi-valued dependency.

C) Fully Functional Dependency

Fully Functional Dependency occurs only in a relation with composite primary key. Fully functional dependency occurs when one or more non key attributes are depending on all parts of the composite primary key.

Table 3.7.3 : Stud_Courses

Stud_Id	Stud_Name	Contact_No	Course_Id	Course_Name	Credit_Hours	Grade
1001	Jasmin	9950043321	C_310	Database Management System	3	A
1001	Jasmin	9950043321	C_420	Operating System	3	B
1002	Smita	9800678120	C_310	Database Management System	3	B
1002	Smita	9800678120	C_420	Operating System	3	B
1002	Smita	9800678120	C_422	Computer NetWork	3	C
1003	Rahul	8878712356	C_412	Software Engineering	3	A

Here the composite primary key is Stud_Id + Course_Id

- Attribute Grade is fully functionally dependent on the primary key (Stud_Id, Course_Id) because both parts of the primary keys are needed to determine Grade.
- On the other hand both Stud_Name, and Contact_No attributes are not fully functionally dependent on the primary key, because only a part of the primary key namely Stud_Id is needed to determine both Stud_Name and Contact_No.
- Also attributes Credit-Hours and Course-Name are not fully functionally dependent on the primary key because only Course-Id is needed to determine their values.

D) Partial Functional Dependency

- **Partial Functional Dependency** occurs only in a relation with composite primary key. Partial functional dependency occurs when one or more non key attributes are depending on a part of the primary key.

- In partial functional dependency the determinant (Left-hand side of FD) consists of key attributes, but not the entire primary key and the determined (Right-hand side of FD) consist of non-key attributes.

For example,

Here the composite primary key is Stud_Id + Course_Id

Stud_Id	Stud_Name	Contact_No	Course_Id	Course_Name	Credit_Hours	Grade
1001	Jasmin	9950043321	C_310	Database Management System	3	A
1001	Jasmin	9950043321	C_420	Operating System	3	B
1002	Smita	9800678120	C_310	Database Management System	3	B
1002	Smita	9800678120	C_420	Operating System	3	B
1002	Smita	9800678120	C_422	Computer NetWork	3	C
1003	Rahul	8878712356	C_412	Software Engineering	3	A

- To determine name of the student or contact number of the student we can use only Stud_Id attribute which is part of the primary key.

{Stud_Id} → {Stud_Name, Conatct_No}

E) Transitive Functional Dependency

SPPU - May 13, Dec. 13, May 15

University Questions

Q. Describe the concept of transitive dependency. **(May 2013, Dec. 2013, 3 Marks)**

Q. Define Transitive dependency. **(May 2015, 2 Marks)**

- Functional dependency is said to be **transitive** if it is indirectly form by using two functional dependencies.
- For example in relation R (A, B, C), Functional dependency F includes:

A → B and B → C both the FDs are hold by relation, and if B → A doesn't hold

- Then we can say that A → C also holds by the relation.
- This type of dependency is called as **transitive dependency.**

Example

If Student_Courses relation holds following FDs:

{Course _Id} → {Student_Name}

{Student_Name} → {Student_Age}

Then we can say that the relation also holds following dependency.

{Course_Id} → {Student_Age}

F) Trivial Functional Dependency

- The dependency of an attribute on a set of attributes is known as trivial functional dependency if the set of attributes contains the attribute itself.
- In this dependency the Right-Hand Side of FD is a subset of Left-Hand Side of the FD. Functional dependency of the form A → B is trivial if B is a subset of A.
- The functional dependencies {A, B} → A is trivial.
- In a relation, if attribute A is depend on a set of attributes AB, and also the attribute A is subset of AB then this functional dependency is said to be trivial.

For example,

Stud_Courses

Stud_Id	Stud_Name
1001	Jasmin
1002	Smita
1003	Rahul

- {Stud_Id, Stud_Name} → {Stud_Id} is a trivial functional dependency as {Stud_Id} is subset of {Stud_Id, Stud_Name}. By knowing the values of id and name of the student it is possible to find values of student_id uniquely.
- The functional dependencies :

Stud_Id → Stud_Id and Stud_Name → Stude_Name

is also referred as trivial dependencies.

G) Non Trivial Functional Dependency

- It is inverse to trivial dependency. Functional Dependency is said to be **non trivial** if none of the attributes of Right-Hand Side of FDs are part of the Left-Hand Side attributes.
- Functional dependency of the form $A \rightarrow B$ is non-trivial if values in B are not present in A i.e. (B is not a subset of A).
- Non-Trivial Functional Dependency can be categorized into:

(i) Complete Non Trivial Functional Dependency

Functional Dependency is **completely non trivial** if none of the Right-Hand Side attributes of FD are part of the Left Hand Side attributes of the FD.

For example,

{Stud_ID} → {Stud_Name}

Values of Stud_Name attribute are totally different from values of Stud_Id in a relation.

(ii) Semi Non Trivial Functional Dependency

A Functional Dependency is **semi non trivial** if at least one of the Right Hand Side attribute of FD is not part of the Left Hand Side attributes of FD.

For example,

{Stud_ID, Student_Name} → {Stud_Name, Contact_No}

Values of Contact_No attribute are totally different from values of {Stud_Id, Stud_Name} in a relation.

3.7.1.2 Closure of Functional Dependency

1. The set of all functional dependencies logically implied by F are known as the closure of set F.
2. The closure of F is denoted by F^+.
3. To compute F^+, we can use some rules of inference like Armstrong's Axioms.

3.7.1.3 Inference Rules for Functional Dependencies

- The **inference rule** is an assertion which is used to derive new FDs from Previously defined basic FDs. A set of new functional dependencies is derived by applying assertions on previously defined basic functional dependency set.
- It is problematic to represent sets of all FDs for a relation initially.
- To make it easy, first define only basic FDs. Then apply set of inference rules on those FDs to derive new set of FDs.
- The most basic inference rule is Armstrongs's Axioms. There are other rules such as union, decomposition and pseudo transitivity which are derived from Axioms.

A) Armstrong's Axioms

SPPU - May 14

University Question

Q. State & prove Armstrong's Axioms rules for functional dependencies ?

(May 2014, 8 Marks)

Armstrong's axioms were developed by William W. Armstrong. These are a set of axioms (or inference rules) which are used to infer (conclude) all the FDs on a relational database.

Consider a relation R having three sets of attributes X, Y, Z. Then the Armstrong's axioms are :

- **Reflexivity rule :** If in a relation R values of Y attribute set are the subset of values in X attribute set, then FD $X \rightarrow Y$ can hold by the relation R. This type of dependency also called as trivial dependency.
- **Augmentation rule :** If FD $X \rightarrow Y$ is hold by relation R, then FD $XZ \rightarrow YZ$ also hold by relation.
- **Transitivity rule :** If Functional dependencies $X \rightarrow Y$ and $Y \rightarrow Z$ are hold by relation R then the FD $X \rightarrow Z$ also followed by relation R. This type of dependency also called as transitive dependency.

Properties of Armstrong's axioms

- The Armstrong's Axioms are **sound** :
  - When basic functional dependencies F on a relation R are given, then the FDs generated by using Armstrong's axioms will hold by R.
  - In other words applying Armstrong axioms on set of FDs which are previously defined will not generate invalid FDs.

- The Armstrong's Axioms are **complete** : When basic functional dependencies F on a relation R are given, then each and every valid Functional dependency on relation R can be found by applying only Armstrong axioms on set of FDs which are previously defined.

B) Union or Additivity

If in relation R having three sets of attributes X, Y, Z, and the set of functional dependencies : X $\rightarrow$ Y and X $\rightarrow$ Z are hold by relation R, then the functional dependency X $\rightarrow$ YZ also hold by relation R. This rule is called as **union.**

C) Decomposition or Projectivity

If in relation R having three sets of attributes X, Y, Z and the functional dependency X $\rightarrow$ YZ is hold by relation R then the following sets of Functional dependencies also hold by relation

R: X $\rightarrow$ Y and X $\rightarrow$ Z

This inference rule is called as **decomposition rule.**

D) Pseudo transitivity

If in relation R having four sets of attributes W, X, Y, Z and the functional dependencies X $\rightarrow$ Y and YW $\rightarrow$ Z are hold by relation R, then the relation also hold following functional dependency XW $\rightarrow$ Z. This inference rule known **pseudo transitivity.**

Syllabus Topic : Algorithm for Decomposition

3.7.2 Decomposition SPPU - Dec. 13, May 14

University Questions

Q. What is decomposition ? **(Dec. 2013, 2 Marks)**

Q. What do you mean by "Decomposition"? What are the desirable properties of it? How can we implement them ? **(May 2014, 8 Marks)**

One solution to eliminate the redundancy and also the update and deletion anomalies in database is breaking a relation into two or more relations. This process is called as **decomposition.**

Let, R be a relation schema.

A set of relation schemas {R1, R2,..., Rn} is a decomposition of R if

$$R = R1 \cup R2 \cup .....\cup Rn$$

For example,

Let us know how to break the relation into more than one relation.

Consider CLASSINFO relation having COURSE_ID, COURSE_NAME, STUD_NAME, SUB_ID, and SUB_NAME.

Table 3.7.4 : CLASSINFO

COURSE_ID	COURSE_NAME	STUD_NAME	SUB_ID	SUB_NAME
C_101	F. E.	Nishant	S_112	C
C_102	S. E.	Prashant	S_212	Java
C_102	S. E.	Prashant	S_222	DBMS
C_103	T. E	Sanvi	S_311	Ad. Java
C_101	F. E	Sanvi	S_115	CPP

This relation can be decomposed into two relations as follows :

i) COURSE_ STUDENT(COURSE_ID, COURSE_NAME, STUD_NAME)

ii) SUBJECT_ STUDENT(SUB_ID, SUB_NAME, STUD_NAME)

Table 3.7.5 : COURSE_STUDENT

COURSE_ID	COURSE_NAME	STUD_NAME
C_101	F. E.	Nishant
C_102	S. E.	Prashant
C_103	T. E	Sanvi
C_101	F. E	Sanvi

Table 3.7.7 : SUBJECT_STUDENT

SUB_ID	SUB_NAME	STUD_NAME
S_112	C	Nishant
S_212	Java	Prashant
S_222	DBMS	Prashant
S_311	VB	Sanvi
S_115	CPP	Sanvi

Now try to join these two relations COURSE_STUDENT and SUBJECT_STUDENT as follows :

Table 3.7.6 : COURSE_STUDENT $\bowtie$ SUBJECT_STUDENT

COURSE_ID	COURSE_NAME	STUD_NAME	SUB_ID	SUB_NAME
C_101	F. E.	Nishant	S_112	C
C_102	S. E.	Prashant	S_212	Java

COURSE_ID	COURSE_NAME	STUD_NAME	SUB_ID	SUB_NAME
C_102	S. E.	Prashant	S_222	DBMS
C_103	T. E	Sanvi	S_311	VB
C_103	T. E	Sanvi	S_115	CPP
C_101	F. E	Sanvi	S_115	CPP
C_101	F. E	Sanvi	S_311	VB

Here two additional rows are displayed this states that the bad design of decomposition may leads to information loss. So, to ensure that the database has good design after decomposing; the decomposition process is done through some criteria which is defined by desirable properties of decomposition.

3.7.2.1 Desirable Properties of Decompositions

While decomposing the relation there must be some properties to hold so that the database remains in consistent state. There are several properties which are required to hold while decomposing the database schema.

There are two properties of decomposition :

i) Lossless join Decomposition

ii) Dependency Preservation Decomposition

i) Lossless decompositions

- When a relation is decomposed into several relations it is essential that the decomposition does not leads to any loss of the information.
- If there is the loss of the information, then it is called lossy decomposition or **lossy-join decomposition**. Decomposition is called as lossless decomposition if there is no loss of information in the decomposition of the relation into the desirable smaller relations.
- A decomposition {R1, R2,..., Rn} of a relation R is called a **lossless decomposition** for R if the natural join of R1, R2,..., Rn produces exactly the relation R.
- Decomposition is lossless if we can recover to original relation from combining smaller decomposed relations as follows :

  Thus, R` = R

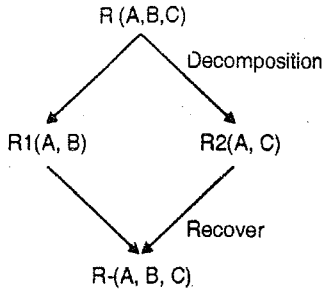


Fig. 3.7.1

Lossless decomposition property

- The lossless join decomposition is defined as, if R is a relation schema and F is a set of functional dependencies on R. R1 and R2 form decomposition of R. If at least one of following functional dependency is hold by F^+ then the decomposition is called as **lossless-join decomposition.**

F :

- $R1 \cap R2 \rightarrow R1$, that is all attributes common to both R1 and R2 functionally determine ALL the attributes in R1 **OR**

  $R1 \cap R2 \rightarrow R2$, that is all attributes common to both R1 and R2 functionally determine ALL the attributes in R2
- In other words, if $R1 \cap R2$ forms a superkey of either R1 or R2. The decomposition of R is a lossless decomposition.

ii) Dependency preservation decomposition

- To ensure that the functional dependencies hold by relation R should also preserved by the smaller relations (R1, R2, ..., Rn), the dependency preservation property of decomposition is used.
- A decomposition of the relation R into smaller relations R1, R2, ..., Rn is dependency preserving if all the Functional dependencies within R are followed in relations R1, R2, ..., Rn.

Dependency preservation decomposition property

- **Dependency preservation decomposition** is defined as, If R is a relation schema and F is a set of functional dependencies on R. R1, R2, ..., Rn form decomposition of R. If

$$(F1 \cup F2 \cup \ldots \cup Fn)^+ = F^+$$

Where,

F1, F2, ... Fn are the functional dependencies of relation R1, R2, ..., Rn.

F is a functional dependency of relation R

$(F1 \cup F2 \cup \ldots \cup Fn)^+$ is a closure of union of all sets of functional dependencies of R1, R2, ...Rn.

F^+ is a closure of set of functional dependency of relation R.

Syllabus Topic : 2NF

3.8 Second Normal Form (2NF)

SPPU - Dec. 15, May 16, Oct. 16

University Questions

Q. Explain 2nd Normal form with suitable example. **(Dec. 2015, 3 Marks)**

Q. Explain Second Form with suitable example. **(May 2016, Oct. 2016(In sem), 3 Marks)**

- To understand the second normal form fist we should get concepts of Prime and Non-prime attributes.
- **Prime attribute :** An attribute, which is a part of the prime-key is known as a prime attribute.
- **Non-prime attribute :** An attribute, which is not a part of the prime-key is said to be a non-prime attribute.
- As per the rule of **Second Normal Form**, every non-prime attribute must be dependent upon the prime attribute.
- If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute.
- Consider Table 3.8.1.

Table 3.8.1 : STUDENT_PROJECT

STUD_ID	PROJ_ID	STU_NAME	PROJ_NAME
S101	P1	Kunal	Banking System
S102	P2	Radhika	Library Management
S103	P3	Kiran	Speech to Text Converter
S104	P4	Jay	ATM

- In the above example the prime key attributes are STUD_ID and PROJ_ID. Here the non key attributes like STU_NAME and PROJ_NAME are dependent upon one of the prime key attributes.
- Means STUD_NAME is depend upon STUD_ID while the PROJ_NAME depends upon PROJ_ID. This is called as partial dependency.
- As per the rule of Second Normal Form the non key attributes should be dependent upon all the key attributes which is not followed.
- To convert the data in Second Normal Form we have to split the table in two different tables as follows.

STUDENT

STUD_ID	STU_NAME	PROJ_NAME
S101	Kunal	Banking System
S102	Radhika	Library Management
S103	Kiran	Speech to Text Converter
S104	Jay	ATM

- In the table STUDENT all the non prime attributes like STU_NAME and PROJ_NAME are completely depends upon the prime attribute STUD_ID

PROJECT

PROJ_ID	PROJ_NAME
P1	Banking System
P2	Library Management
P3	Speech to Text Converter
P4	ATM

- In the table PROJECT the non prime attributes PROJ_NAME is completely depends upon the prime attribute PROJ_ID

Syllabus Topic : 3NF

3.9 Third Normal Form (3NF)

SPPU - May 13, Dec. 13, May 15, May 16, Oct. 16

University Questions

Q. Explain how this concept is used to define 3 NF. **(May 2013, Dec. 2013, 5 Marks)**

Q. Explain third normal form with suitable example.
(May 2015, May 2016, Oct. 2016(In sem), 3 Marks)

A database design is said to be in **3NF** if both the following conditions are satisfied by it

- Initially the design must be in 2NF.
- No non-prime attribute should be transitively dependent on prime key attribute.
- For any **non-trivial** functional dependency, $X \rightarrow A$

  Either X is a superkey or A is prime attribute.
- Consider the Table 3.9.1.

Table 3.9.1 : STUDENT_DETAILS

STUD_ID	STU_NAME	CITY	ZIP
S101	Kunal	Pune	411037
S102	Radhika	Nasik	422001
S103	Kiran	Mumbai	400016
S104	Jay	Nagpur	440001

- In Table 3.9.1 the STUD_ID is the only one primary key. Here the data of city can be retrieved through either STUD_ID or ZIP code. But the CITY is not superkey as well as nor the CITY is prime attribute.
- Here the CITY is depends upon ZIP and ZIP is depends upon STUD_ID
- Stud_id → zip → city.
- This relationship is known as **transitive dependency**.
- We have to remove this transitive dependency by implementing Third Normal Form. For this purpose we have to split the STUDENT_DETAILS table in two different tables say STUDENT_DATA and CITY.

STUDENT_DATA

STUD_ID	STU_NAME	CITY
S101	Kunal	Banking System
S102	Radhika	Library Management
S103	Kiran	Speech to Text Converter
S104	Jay	ATM

CITY

ZIP	CITY
411037	Pune
422001	Nasik
400016	Mumbai
440001	Nagpur

- Now the database design is converted into Third Normal Form. Here the basically design is already in Second Normal Form and No non-prime attribute is transitively dependent on prime key attribute.

Syllabus Topic : BCNF

3.10 Boyce-Codd Normal Form (BCNF)

SPPU : May 14

University Question
Q. Define BCNF. **(May 2014, 2 Marks)**

A database design is said to be in BCNF if both the following conditions are satisfied by it

- Initially the table must be in 3NF.
- For any non-trivial functional dependency $X \rightarrow A$, X must be a super-key.

In above database design, Stud_id is the super-key in the relation Student_Data and Zip is the super-key in the relation City

Stu_ID → Stu_name, City

And Zip → City

Which confirms that both the relations are in BCNF.

3.10.1 Difference between 3NF and BCNF

SPPU : May 14

University Question
Q. Differentiate between BCNF & 3NF. How it is stronger than 3NF? **(May 2014, 6 Marks)**

Sr. No.	3NF	BCNF
1.	It stands for Third Normal Form.	It stands for Boyce-Codd Normal Form.
2.	A database design should be already in 2NF to convert in 3NF.	A database design should be already in 3NF to convert in BCNF.

Sr. No.	3NF	BCNF
3.	Rule : For any non-trivial functional dependency, X → A Either X is a superkey or A is prime attribute	Rule : For any non-trivial functional dependency, X → A X must be a super-key.
4.	3NF is weaker than BCNF.	BCNF is stronger than 3NF.
5.	3NF cannot catch all the anomalies.	BCNF was developed to capture those anomalies that could not be captured by 3NF.
6.	More redundancy.	Less redundancy.
7.	Computational time is more.	Computational time is less.

The above difference clearly indicates that **BCNF is stronger than 3NF.**

3.10.2 BCNF Decomposition Algorithm

Input : A relation R and functional dependencies F which are hold on R.

Output : A decomposition of R into a set of relations (R1, R2, R3,...., Rn), all of which are in BCNF.

1. If (R is in BCNF) then
   Return R.
2. Else If (there are BCNF violation, let's consider one FD as X→Y) then
3. Begin
4. Compute X^+
5. Decompose R into (R1 and R2) Select $R1=X^+$, and let R2 have attributes X and those attributes of R which are not in X^+
6. Compute the sets of FDs S1 and S2 for R1 and R2, respectively.
7. End
8. Repeat steps from 1 to 5 for R1 and R2 until all the relations (Ri) are in BCNF.
9. Return the union of the result of these compositions.

Example

As follows : If a relation R (A, B, C, D, E) and Functional dependency set F contains

```
F:=
{
   A→BC
   C→DE
}
```

Step 1 : Compute closure of F

```
F+ :=
{
   A+ →ABCDE
   B+ →B
   C+ →CDE
   D+ →D
   E+ →E
}
```

So it concludes that the candidate key is A for relation R

Step 2 to 4 : check whether the FD is in BCNF if not decompose it.

Again consider the functional dependency set F :

First FD : A→BC, has candidate key A at the LHS of FD so it is in BCNF

Second FD : C→DE, doesn't contain LHS as candidate key so it violates the rule of BCNF. So let's decompose it to make the FD in BCNF, into R1 and R2 where R1 contains attributes contained in FD that is in BCNF and R2 contains attribute which are in FDs that violates BCNF rule.

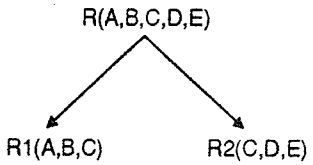


Fig. 3.10.1

Now R1 has set (ABC) which contain attribute A as a key according to the A^+ closure. And R2 has a set (CDE) which also contain attribute C as key According to the C^+ closure. At the end both decomposed relations (R1 and R2) are in BCNF form.

3.11 Fourth Normal Form

SPPU - May 13

University Question

Q. Explain why 4NF is more desirable than BCNF. Rewrite the definition of 4NF and BCNF using the notions of domain constrains and general constraints? **(May 2013, 8 Marks)**

In the Fourth Normal Form,

- Initially the design must be in 3NF.
- It should not contain any multivalued dependencies.

Consider Table 3.11.1. It contains the data about Subject, Lecturer and Book_Name referred by the lecturers.

Table 3.11.1

Subject	Lecturer	Book_Name
English	Prof. Jitesh	English Book 1
English	Prof. Meetali	English Book 2
Science	Prof. Patil	Science Book 1
Maths	Prof. Vinay	Maths Book 1

- Now this design satisfies the rules of 3NF but two attributes Lecturer and Book_Name are independent entities.
- The subject English can teach by either Prof. Jitesh or Prof. Meetali. And also student have two choices for books of English. They can either refer English Book 1 or English Book 2.
- Hence the relationship will be

  SUBJECT → LECTURER

  SUBJECT → BOOK_NAME
- The relationship shows multi-valued dependency of attribute SUBJECT.
- If we need to select both lecturer and books recommended for any of the subject, then it will show the combination of lecturer, books, which implies lecturer who recommends which book. This is not correct.
- To remove this dependency the table should be split into two tables as below :

Lecturer

Subject	Lecturer
English	Prof. Jitesh
English	Prof. Meetali
Science	Prof. Patil
Maths	Prof. Vinay

Book

Subject	Book_Name
English	English Book 1
English	English Book 2
Science	Science Book 1
Maths	Maths Book 1

- Now in this design, if we want to retrieve names of lecturer or books recommended, then two independent queries can be executed. This eliminates the multi-valued dependency of attribute "Subject". Hence the design is in 4NF.

3.11.1 Difference between 4NF and BCNF

Sr. No.	4NF	BCNF
1.	It stands for Fourth Normal Form.	It stands for Boyce-Codd Normal Form.
2.	A database design should be already in 3NF and BCNF to convert in 4NF.	A database design should be already in 3NF to convert in BCNF.
3.	No multi-valued dependencies exist in the tables.	Multi-valued dependencies exist in the tables.
4.	The 4NF is more desirable than BCNF because it avoid repetition of data.	The BCNF is less desirable than BCNF because it does not avoid repetition of data.
5.	The decomposition in 4NF does not lead to loss of information when lossless join decomposition is used.	This is little bit difficult in BCNF.
6.	Redundancy is less.	Redundancy is more.

Syllabus Topic : Modeling Temporal Data

3.12 Modeling Temporal Data

- In conventional databases system, the time period is not attached to the data. The data is considered as valid for now. They do not keep track of past or future database states.
- **Temporal data** is the data to which time period is attached to it. This time period indicates that when that data is valid or stored in the database. By attaching a time period to the data, it becomes possible to store different states of the database.
- To implement the temporal database first we have to timestamp the data. This allows the distinction of different database states.
- One approach is that we can timestamp the entities with periods and another approach is that we can timestamp the property values of the entities.
- Tuples or rows are timestamped in the relational data model, while objects and/or attribute values may be timestamped.in object-oriented model.
- How and what time period are stored in databases? In general two different notions of time which are relevant for temporal databases are stored in database. First is called the valid time, the second is called as the transaction time.
- Valid time is nothing but the time period in which a fact is true with respect to the real world. Transaction time is the time period in which a fact is stored in the database.
- These two time periods do not have to be the same for a single fact. Consider that we are storing data about cricket matches played in 2016. Then the valid time of these facts is somewhere between Jan 01 2016 and Dec. 31 2016, where as the transaction time starts when we insert the facts into the database, for example, March 23 2017.
- Assume we would like to store data about our employees with respect to the real world. Then, the format of table could be:

Table 3.12.1 : EMP_DETAILS

Empid	Name	Department	Salary	Valid_time_start	Valid_time_end
101	Rohit	Production	11000	2013	2015
101	Rohit	Sales	13000	2015	2016
102	Vinay	Research	13000	2011	2015
103	Sunil	Accounts	10500	2014	INF

- Table 3.12.1 we have stored the history of the employees with respect to the real world.
- The above valid-time table stores the history of the employees with respect to the real world. The attributes VALIDTIMESTART and VALIDTIMEEND actually represent a time interval which is closed at its lower and open at its upper bound.
- Thus, we see that during the time period [2013 – 2015], employee Rohit was working in the Production department, having a salary of 11000. Then he changed to the sales department with salary raise to 13000.
- The upper bound INF indicates that the tuple is valid until further notice. Storing information about past states is possible now.
- We see that Vinay was employed from 2011 until 2015. In this case, in non-temporal table, this information was (physically) deleted when Vinay left the company.

3.12.1 Different Forms of Temporal Databases

- There are forms of temporal databases which depend upon the two different notions of time - valid time and transaction time.
- **Historical database** stores data with respect to valid time.
- **Rollback database** stores data with respect to transaction time.
- **Bitemporal database** stores data with respect to both valid time and transaction time.
- Usually in the commercial DBMS only a single state of the real world is stored, in general the most recent state. Such databases are known as **snapshot databases**.
- In the context of valid time and transaction time, the snapshot database is denoted in the Fig. 3.12.1.

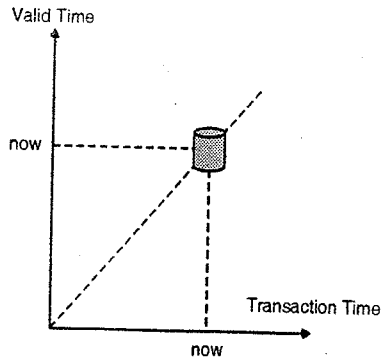


Fig. 3.12.1

- In case of bitemporal database the history of data related to both valid time and transaction time is stored in the database. TimeDB is an example of temporal database.
- In TimeDB DBMS the table may be in following formats :
  - Snapshot table - stores only current data
  - Valid-time table - stores when the data is valid with respect to the real world
  - Transaction-time table - stores when the data was recorded in the database
  - Bitemporal table - stores both valid time and transaction time
- In the SQL extended version it is also specified that which kind of table is needed when the table is created. It is also possible to make changes in existing table which is known as schema versioning.
- The temporal queries, temporal constraints and temporal modification statements are also supported by it.
- Fig. 3.12.2 specifies the different states of bitemporal database.

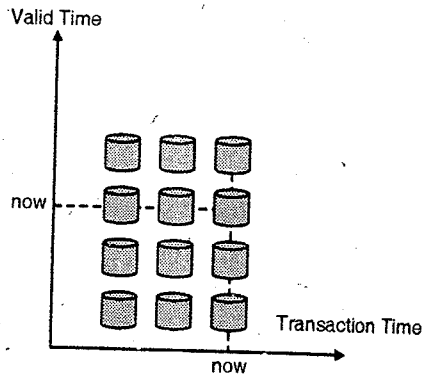


Fig. 3.12.2

□□□

UNIT IV

CHAPTER 4

Database Transactions and Query Processing

Syllabus

- Basic concept of a transaction
- Transaction Management
- Properties of Transaction
- Concept of Schedule,
- Serial schedule,
- Serializability : Conflict and View, Cascaded Aborts
- Recoverable and Non-recoverable schedules
- Concurrency Control : Need, Locking Methods, Deadlocks, Time-Stamping Methods
- Recovery Methods: Shadow-paging and Log-Based Recovery, Checkpoints
- Query Processing, Query Optimization,
- Performance Tuning

Syllabus Topic : Basic Concept of Transaction

4.1 Basic Concept of Transaction

4.1.1 Transaction

SPPU : May 13, Dec. 13, May 14

University Question

Q. Explain the concept of 'transaction'.
(May 2013, Dec. 2013, May 2014, 4 Marks)

- A **transaction** is a series of operations performed as a single logical unit of work on the Database Management System. Transaction leads to modification in the database contents.
- A transaction is initiated by a user program written in the high level data manipulation languages like SQL or programming language like java. The transaction consists of all the operations executed between begin transaction and end transaction.
- Transactions in a database environment have two main purposes :
  (1) To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.

  For example, in a baking system, transferring money from one account to another leads to change in balance values of both accounts. To keep the database in consistent state both changes must succeed or fail together.

  (2) To provide isolation between programs accessing a database concurrently. If this isolation is not provided, the programs outcomes are possibly incorrect.

Process of Transaction

The execution of transaction leads to perform series of read and write operations on database objects, which are explained below:

(A) Read Operation

To perform transaction on a data which is present in main memory Read ( ) operation is used.

Read (X);

This statement states that value of X is read from main memory (i.e. from local buffer).

If it is not present in the main memory then:

(1) It is first brought into main memory from disk, using Input ( ) operation.

Like **Input (X);**

(2) Then its value is copied into a program variable as shown in Fig. 4.1.1.

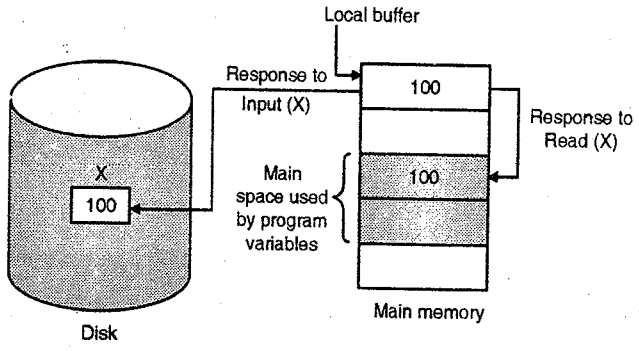


Fig. 4.1.1

(B) Write Operation

To write a database object, an in-memory copy of the object is first modified and then written to disk.

Write (X);

This statement states that value of X is written to local buffer which is in main memory. And then this updated value is copied into appropriate disk block, so that the changes are done permanently as shown in Fig. 4.1.2.

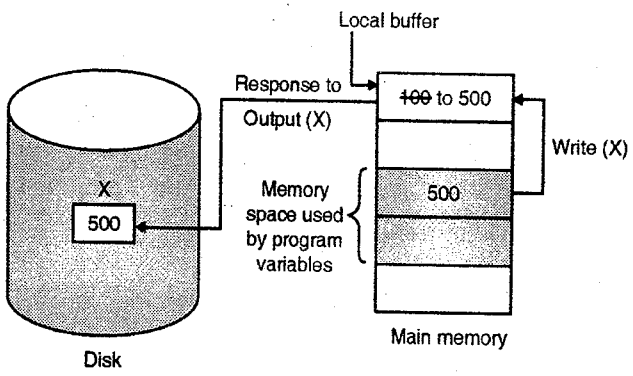


Fig. 4.1.2

Syllabus Topic : Properties of Transaction

4.2 Properties of Transaction

SPPU - May 13, Dec. 13, May 14

University Question

Q. Describe ACID properties for transaction. **(May 2013, Dec. 2013, May 2014, 4 Marks)**

In computer science, ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties of database transactions. In the context of databases, a sequence of database operations that satisfies the ACID properties and thus, can be perceived as single logical operation on the data is called a **transaction**.

ACID properties are explained in detail as follows :

(A) Atomicity

- Atomicity is based on the concept that each transaction be "all or nothing" : if any one part of the transaction in a sequence fails, then the entire transaction fails, and there will be no change in database state.
- An atomic system must guarantee atomicity in every situation, including power failures, errors and crashes. For the outside world, a committed transaction appears completely by its effects on the database. That means it should be indivisible ("atomic"), and an aborted transaction does not happen.
- This property states that, either all operations contained by a transaction are done successfully or none of them complete at all. To maintain the consistency of data this property is very useful.

(B) Consistency

- The consistency property ensures that the transaction executed on the database system will bring the database from one valid state to another.
- The data which is written by the transaction must be valid according to the all standard rules and regulations regarding constraints, cascades, triggers etc. Consistency does not guarantee accuracy of the transaction as per the expectations of programmer.

(C) Isolation

- When transactions are performed in a sequence, the state of a system is always valid without any problem. But sometimes we may have to perform multiple transactions concurrently.
- In case of concurrent transactions, the isolation property ensures that the system state should be same that would be obtained if transactions were executed sequentially, i.e., one after the other. The effect of any incomplete transaction should be invisible to other transaction by means of isolation.

(D) Durability

- The durability property assures that after transaction committed successfully the updates made should remain permanent in the database even in the event of power loss, crashes or errors.
- For example in a database management system, when a series of transaction is executed, the modification done should be stored permanently even if the database crashes just after the transaction. To avoid problem because of power loss, the states of database after every transaction must be recorded in a non-volatile memory.

A complete example of transaction

- Consider a banking system having number of accounts stored in the database.
- To access the database following operations are used in transaction:
- Read(X), which transfer data X from database to local buffer & Write(X), which transfer the data X from local buffer to database.
- Suppose, before the transaction M has \$500 and N has \$1000 balance in their accounts.
- Let T_i be a transaction which will transfer \$150 from account M to N, this will be written as :

```
Ti   :   Read(M);
         M := M – 150;
         Write(M);
         Read(N);
         N := N + 150;
         Write(N);
```

- **Atomicity :** If any failure occurs in the system during transaction proceeded up to Write (M); and further operations does not executed due to failure occurrence, it will leads the banking database system in inconsistent state if atomicity property is not provided.
- But if the atomicity property is provided then if all operations are not executed, then the operations which are executed before failure get cancelled.
- **Consistency :** If consistency is provided, then in above example the sum of balance in account M and N will be same before transaction Ti perform and after Ti completes. That is, in initial situation M + N gives \$1500. If transaction committed successfully then M holds \$350 and N holds \$1150. Addition of amount will give \$1500. If transaction does not committed successfully then M holds \$500 and N holds \$1000. Addition of amount will give \$1500.
- **Isolation :** This property takes care of no other transactions should interfere with transaction Ti during its execution.
- **Durability :** It ensures that the data modification done by transaction Ti will persist in database even if any failure occurs during system after committing the transaction.

4.3 Transaction States SPPU - Oct. 16

University Question

Q. List and explain different states through which transaction goes during its execution.
(Oct. 2016 (In sem), 5 Marks)

There are five states of transaction :

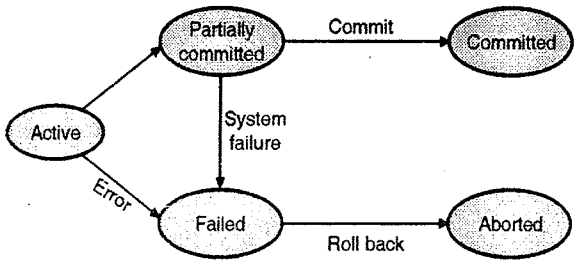


Fig. 4.3.1 : Transaction states

During execution, transaction will be in one of the following state.

Active

- This is the first state of transaction. In this state the transaction is being executed. This state is entry point to every transaction. Transactions

which are in active state indicate that their execution has been started.

Partially Committed

- When a transaction completes all operations, it will enter into partially committed state. Even the last operation has been performed; data is still not saved to the database.

Failed

- If execution of transaction cannot proceed due to failure of the system or failure in database, then the transaction is said to be in failed state.

Abort

- In case of the failed transaction, the modification done in database during transaction processing must be rolled back to ensure the atomicity and consistency of database.
- The transaction enters into 'Abort' this state after roll back operation is performed. It is the end of transaction when any fault occurs.

Committed

- If without any error transaction completed successfully it will come into committed state which will allow to made changes permanent into database. This is the last step of a transaction, if it executes without fail.
- Every transaction goes through at least three states among of five. Either transaction may goes through active -partially committed - committed or through active – Failed - aborted, or through active - partially committed – failed - abort.

Syllabus Topic : Transaction Management

4.4 Transaction Management

To ensure database integrity it is important to control transactions. Transactions can be controlled using TCL commands which we have studied in previous Chapter 2.

Operations performed on transactions

Commit and roll back operations are performed on transaction to manage transaction.

- **Commit** operation allow transaction to update database permanently. It will ensure the updates done by transaction will persist even if failure occurs. Commit operation leads the transaction into committed state.
- **Rollback** operation helps to maintain the atomicity property of transaction. If any failure occurs during transaction processing which leads the transaction to failed state. In this situation roll back operation cancels the changes made by that transaction in database to maintain consistency. And then the transaction will be aborted.

In general three utilities are used to manage transactions.

(1) Transaction manager

- Ensures that database proceed from one consistent state to another consistent state.
- Ensures that the transaction should not violet integrity constraints.

(2) Scheduler

- Provides a specific strategy for the execution of transaction and corresponding concurrency control.
- Avoid or resolves conflicts during concurrent data access.

(3) Recover manager

Restore the state of database it was in before the failure of transaction.

Syllabus Topic : Concept of Schedule

4.5 Concept of Schedule

4.5.1 Schedule

- A **schedule** is a collection of several transactions that can be executed in a predefined order as a single unit.
- Schedule may combine the operations of one transaction with other transaction. These operations must be in same sequence as they occur in their individual transactions.
- Schedule is necessary in a situation when a system has several transactions which are going to be performed on same database object. If those

transactions are executed in parallel, they may affect the result of the transaction.

For example

- Consider transaction T_1 and T_2 are present in the system, and both use the values of account A commonly. If one transaction is updating the values which the other transaction is accessing, then the order of these two transactions will change the result of second transaction. Hence a schedule is created to execute the transactions.
- Depending on the arrangement of these transactions, schedule can be divided into two types : Serial schedule and concurrent schedule.

Syllabus Topic : Serial Schedule

4.5.2 Types of Schedule

(A) Serial Schedule

- In serial schedule all the transactions in the schedule are executed one after another. Operations of different transactions are not interspersed with each other.
- For example, account M has \$500 and account N has \$1000 initially. Transaction T1 transfers \$200 from account M to N and Transaction T2 transfers \$100 from account M to N. T_1 and T_2 can be scheduled in two ways for execution:

Schedule 1 : Serial schedule for T_1 followed by T_2

T_1	T_2
Read(M);	
M=M-200;	
Write(M);	
Read(N);	
N=N+200;	
Write(N);	
	Read(M);
	M=M-100;
	Write(M);
	Read(N);
	N=N+100;
	Write(N);

After execution of both transactions M holds \$200 and N holds \$1300.

Schedule 2 : Serial schedule for T_2 followed by T_1

T_1	T_2
	Read(M);
	M=M-100;
	Write(M);
	Read(N);
	N=N+100;
	Write(N);
Read(M);	
M=M-200;	
Write(M);	
Read(N);	
N=N+200;	
Write(N);	

After execution of both transactions M holds \$200 and N holds \$1300.

(B) Concurrent Schedule

Concurrent schedule is a schedule where, several transactions are executed simultaneously. The operations of one transaction can be mixed with operations of other transaction.

Schedule 3 : Concurrent schedule for T_1 and T_2

T_1	T_2
Read(M);	
M=M-200;	
Write(M);	Read(M);
	M=M-100;
	Write(M);
Read(N);	
N=N+200;	
Write(N);	
	Read(N);
	N=N+100;
	Write(N);

After execution of both transactions M holds \$200 and N holds \$1300.

In this example concurrent schedule also provides same result as the above two serial schedule provides. But not all the concurrent schedules provide result in consistent state.

4.5.3 Advantages of Concurrent Execution of Transactions

Though it is difficult to ensure the isolation while executing transactions concurrently, it has different advantages as follows :

- **Reduced waiting time**

  System has several transactions which may be short or long. In serial execution of transaction, short transactions need to wait more when long transactions are being processed. If these transactions executed concurrently, it will reduce the waiting time.

- **Increased average response time**

  In concurrent execution, the multiple transactions executes concurrently by sharing various system resources, the waiting time get decreased because of which the response time automatically get increased.

- **Improved resource utilization**

  Number of transactions run concurrently in the system, in which if any transaction needs to wait for some operations to complete, the CPU has ability to switch to another transaction, so the CPU can be utilized in efficient way. It will no more in ideal state.

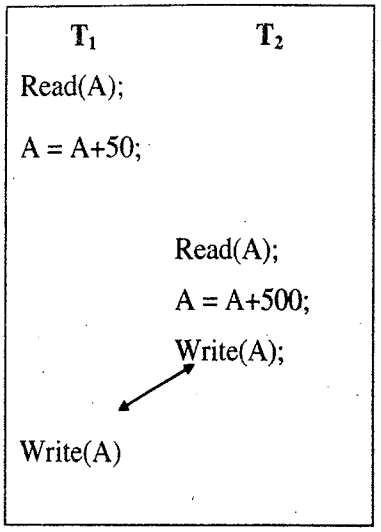
- **Throughput**

  Throughput indicates the number of transaction processed in a time unit. As several transactions get executed simultaneously in the system. Due to efficient resource utilization it will produce their outputs in less amount of time. So numbers of transaction completed is more as compare to serial execution of transactions which will increase the throughput of the system. This leads to increase in system performance.

4.5.4 Problems Occur in Concurrent Execution of Transaction

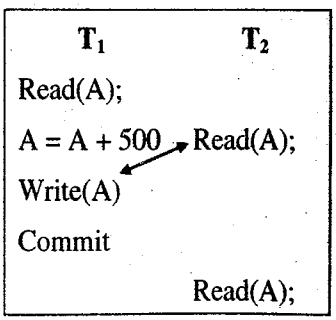
The problem occurs when two transactions are dealing with same data item if one of the transactions performs write operation. If any transaction contains following situations then the conflicts are occurred:

(A) Write-Write conflicts



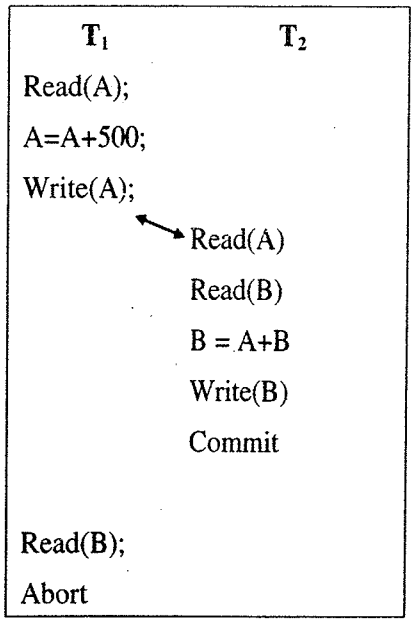
- Here both the transactions T_1 and T_2 read initial values of data item (A) [assume A=1000].
- Initially T_2 adds 500 to A and write new value 1500 of A into database.
- Now T_1 update the initial value of A(1000) to 1050 and write it in the database.
- In between these transactions, the value updated by transaction T_2 is lost.
- Thus write-write conflicts affects lost update on database.

(B) Read-Write conflicts



- Transaction T_1 and T_2 reads initial value of A (1000).
- T_1 updates value of A to 1500 by adding 500 to it and writes it into database and commit.
- Now T_2 again read the value of A which is now 1500.
- Here transaction T_2 reads two different values of same data element A.
- This is read-write conflict.

(C) Write-Read conflicts



- Transaction T1 reads value of A(1000) from database. Add 500 to it and write new value 1500 in the database.
- Transaction T2 reads updated value of A(1500) from database, add it into B and writes value of B in the database.
- But transaction T1 get aborted due to failure and all changes are rolled back.
- Here value read by T2 is dirty read i.e. value updated by uncommitted transaction.

Syllabus Topic : Serializability - Conflict and View

4.6 Serializability : Conflict and View

4.6.1 Serializability

SPPU : May 16

University Question

Q. What is Serializable schedule? **(May 2016, 1 Mark)**

If execution of all the transactions in a concurrent schedule produces same result as in serial schedule then it is called as **Serializable schedule**. **Serializability** is a property of a transaction schedule which is used to keep the data in a consistent state. It relates to the isolation property of a database transaction.

In previous section we have seen which problems may occur in concurrent execution of transaction. Hence, Serializability is must while selecting the concurrent option for execution of transactions.

4.6.2 Difference between Serial Schedule and Serializable Schedule

SPPU - Dec. 14, May 15

University Question

Q. Explain distinct between the terms serial schedule and serializable schedule with suitable example. **(Dec. 2014, May 2015, 5 Marks)**

1. A schedule is said to be Serial if all the operations of transaction are executed serially without any interference from other transaction. For example, schedule 'S1' is a serial schedule if in it only one transaction executes at any given time. When a non-serial schedule produced the same result as serial schedule then it is known as Serializable schedule.
2. The main difference between serial schedule and Serializable schedule is that Concurrency is not allowed in serial schedule that means multiple operations cannot be performed at a time while in serializable schedule concurrency is allowed.
3. If a serial schedule includes only two transactions let's consider T_1 and T_2, then There are only two possible orders to schedule those transactions. First is execute all the operations in transaction T_1 and then executes all the operations in transaction T_2 (i.e. T_1 followed by T_2). The second order is to execute all the operations of transaction T_2 and then execute all the operations of transaction T_1 (i.e. T_2 followed by T_1). But In Serializable Schedule, if it includes two transactions then their operations are mixed (can be executed in parallel) with each other, then there will be many possible orders in which the system can execute the individual operations of the transactions.
4. As the Serial schedule doesn't allow any concurrent execution of transactions the result of the schedule always in consistent state. But in a Serializable schedule the concurrent execution of schedule should be equal to any serial schedule so that the results of schedules are always in consistent state.

5. For example,

Let's considered a schedule S :

T_1	T_2
Read(A)	
Write(A)	
	Read(A)
Read(B)	
Write(B)	
	Read(B)

Serial Schedule

Only two serial schedules are possible for given schedule S as follow:

Serial schedule S1

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(A)
	Read(B)

Serial schedule S2

T_1	T_2
	Read(A)
	Read(B)
Read(A)	
Write(A)	
Read(B)	
Write(B)	

Serializable Schedule

Given schedule S says that transaction T_2 reads updated value of A and reads value of B before modifying it by transaction T_1.

Let us consider 3 schedules S3, S4, and S5. We have to check whether they are Serializable with schedule S or not ?

Schedule S3

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
	Read(A)
	Read(B)
Write(B)	

In schedule S3, transaction T2 reads updated value of A and reads value of B before modifying it by transaction T1. This is same as schedule S. Hence, we can say that the schedule S3 is Serializable with schedule S.

Schedule S4

T_1	T_2
	Read(A)
Read(A)	
Write(A)	
	Read(B)
Read(B)	
Write(B)	

In schedule S4, transaction T2 doesn't read updated value of A. It is not Serializable with schedule S.

Schedule S5

T_1	T_2
	Read(A)
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(B)

In schedule S5, transaction T2 doesn't read updated value of A and it read's value of B which is updated by T1. So it is not serializable with schedule S.

Now in brief we will see the different between Serial and Serializable Schedule.

Sr. No.	Serial Schedule	Serializable Schedule
1.	In serial schedule, transactions are executed one after other.	In serializable schedule, transaction are executed concurrently.
2.	At a time, only one transaction is allowed.	At a time, multiple transactions can be executed at a time.
3.	Resultant database is always consistent.	Care should be taken, to make resultant database consistent.
4.	Takes more time.	Processing is fast.
5.	Suitable if number of transactions are less.	Suitable if number of transactions are more.

4.6.3 Types of Serializability

SPPU : May 14, May 16

University Questions

Q. Explain view and conflict serializability with suitable example. **(May 2014, 8 Marks)**

Q. Explain conflict and view serializable schedule. **(May 2016, 4 Marks)**

Two types of serializability are following

1) Conflict Serializability
2) View Serializability

4.6.3.1 Conflict Serializability

Let's say we have a schedule S and we can reorder the instructions in it and create 2 more schedules S1 and S2.

1. Conflict equivalent schedules

- If a schedule S can be transformed into a serial schedule S' by performing series of swaps of non-conflicting operations, we can say that schedule S & S' are conflict equivalent.
- The schedule S1 and S2 are said to be conflict-equivalent if they satisfies following conditions :
  - Same set of transactions (including sequence *of operations* within each transaction) are involved in both schedules S1 and S2.
  - Order of pair of conflicting-operations is same in both the schedules.

2. Conflicting operations

- Two operations belong to different transactions are said to be conflicting if they performed on same data item and At Least one of them is a write operation.
- **Example**
  - Suppose if transaction T_1 has $Read_1(A)$ operation and transaction T_2 has $Write_2(A)$ operation, then this pair($Read_1(A)$, $Write_2(A)$) is called **Conflicting** operations pair because they belong to two different transactions on same data item A and one of them is write operation.
  - Similarly, ($Write_1(A)$, $Write_2(A)$) and ($Write_1(A)$, $Read_2(A)$) pairs are also know as **conflicting pairs**.
  - On the other hand, ($Read_1(A)$, $Write_2(B)$) pair is **non-conflicting** because they operate on different data items.
  - Similarly, (($Write_1(A)$, $Write_2(B)$) pair is **non-conflicting.**

Ex. 4.6.1 SPPU : Oct. 2016(In Sem), 5 Marks

Check whether following schedules are conflict equivalent or not. Justify your answer. (T_1 & T_2 are transactions)

Schedule S1

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(A)
	Write(A)
	Read(B)
	Write(B)

Schedule S2

T_1	T_2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

Soln. :

Schedule 1

T_1	T_2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(A)
	Write(A)
	Read(B)
	Write(B)

Schedule 2

T_1	T_2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

- In **schedule S1** which is serial transaction,
  - **Transaction T_1**

    First we are performing operations read and write on A and then B
  - **Transaction T_2**

    The same read write operations are performed on A and then B
- In **Schedule S2** which is concurrent transaction,
  - **Transaction T_1**

    First we are performing operations read and write on A
  - **Transaction T_2**

    The same read write operations are performed on A.

    Then again
  - **Transaction T_1**

    Then we are performing operations read and write on B
  - **Transaction T_2**

    The same read write operations are performed on B.
- In schedule S1 (Write(A), Read(A)) and (Write(B), Read(B)) are conflicting pairs.
- As given in example, schedule S2 also follows same order of conflicting pairs as given in schedule S1.
- Here the non conflicting operations can be swapped which does not make any difference to the final result.
- Hence these two schedules are conflict equivalent.
- In other words, transaction T2 of Schedule S1 reads the modified value of A and B.
- Now in second concurrent schedule, when transaction T1 completes both the operations on A, then transaction T2 perform operations on the data item A. because of which no irrelevant output can be generated. The same thing is applicable for data item 'B'.
- So we can say that the Schedule S1 is conflict equivalent to Schedule S2.

A. Conflict serializability

It is defined by equivalence to a serial schedule (no overlapping transactions) with the same transactions, such that both schedules have the same sets of respective chronologically ordered pairs of conflicting operations (same precedence relations of respective conflicting operations).

B. Conflict serializable schedule

Schedule S is conflict serializable if it is conflict equivalent to a serial schedule. The concept of conflict equivalence leads to the concept of conflict serializability, we can say that the schedule S in conflict serializable if it is conflict equivalent to a serial schedule.

Consider the following schedule : S1

T1	T2
$Read_1(A)$	
$Write_1(A)$	
	$Read_2(A)$
	$Write_2(A)$
$Read_1(B)$	
$Write_1(B)$	
	$Read_2(B)$
	$Write_2(B)$

- Possible serial schedules for S1 are :

  T1 → T2 (i.e. all the operations of T1 will perform first and then all the operations of T2 will perform), and T2 → T1 (i.e. all the operations of T1 will perform first and then all the operations of T2 will perform).

- While transferring a schedule into another one following rule must be follow :

  If Oi and Oj are two operations in a transaction and Oi< Oj (Oi is executed before Oj), same order will follow in new schedule as well.

- **Swapping non-conflicting operations** $Read_2(A)$ and $Read_1(B)$ in S1, the schedule becomes,

Schedule S1

T1	T2
$Read_1(A)$	
$Write_1(A)$	
$Read_1(B)$	
	$Read_2(A)$
	$Write_2(A)$
$Write_1(B)$	
	$Read_2(B)$
	$Write_2(B)$

Similarly,

- **Swapping non-conflicting operations** $W_2(A)$ and $W_1(B)$ in S1, the schedule becomes

Schedule $S1_2$

T1	T2
$Read_1(A)$	
$Write_1(A)$	
$Read_1(B)$	
$Write_1(B)$	
	$Read_2(A)$
	$Write_2(A)$
	$Read_2(B)$
	$Write_2(B)$

- $S1_2$ is a serial schedule in which all operations of T1 are performed before starting any operation of T2. Since the S_1 has been transformed into a serial schedule $S1_2$ by swapping non-conflicting operations of S1. So, S1 is **conflict serializable**.

4.6.3.2 View Serializability

- View serializability of a schedule is defined by equivalence to a serial schedule (no overlapping transactions) with the same transactions, such that respective transactions in the two schedules read and write the same data values ("view" the same data values).
- The schedules S1 and S2 are view equivalent if they follows following conditions :
  - In schedule S1, for data item A if read operation is performed by transaction Ti, then in schedule S2, transaction Tj must perform read operation on data item A.
  - If in schedule S1, transaction Ti reads the data written by Tj, then the schedule S2, must have the read operation in transaction Ti, who reads the data written by Tj.
  - If transaction Ti writes the last value of Data item A in schedule S1 then, in schedule S2 last write operation on data item A must be performed by Transaction Ti.
- First two conditions ensure that no conflict occurs during reading the values of data item 'A' in both the schedules. The last condition ensures that the final result of both the schedules is same.
- The concept of view equivalence leads to view serailizability. The schedule S1 is view serializable if it is view equivalent to a serial schedule.

T_1	T_2	T_3
read(A)		
	write(A)	
write(A)		
		write(A)

Schedule S

- Here the Schedule S is view equivalent to the serial schedule < T_1,T_2, T_3>, because the read(A) reads the initial value of A in both schedules and T_3 performs write(A) in both schedules.
- Every view Serializable schedule is not a conflict Serializable schedule, but every conflict Serializable schedule is known as view Serializable schedule.

Syllabus Topic : Recoverable and Non-recoverable Schedules

4.7 Types of Schedules Based on Recovery

4.7.1 Recoverable Schedule

SPPU : May 14, Dec. 16

University Questions

Q. Explain recoverable schedules. **(May 2014, 4 Marks)**

Q. What is recoverable schedule? Why recoverability of schedule is desirable? **(Dec. 2016, 5 Marks)**

If any schedule S is getting correct initial value of the data item after recovering it from failure is called as **recoverable schedule**. The database remains consistence in this schedule.

Consider two transactions T_1 and T_2. Recoverable schedule has some rules as follows :

- In transaction schedule S, for each pair of transaction T_1 and T_2, is schedule such that T_2 can read the data item which is previously written by T_1.
- Commit operation of T_1 must occur before the commit operation of T_2.
- If the rules are not followed then it is difficult to recover the schedule from inconsistent state.

Consider an example

T_1	T_2
Read(X)	
Write(X)	
	Read(X)
Read(Y)	

Suppose after T_2 Read(x) operation, it commits. Then somehow T_1 fails. So, the transaction T_2 must be aborted so as to ensure atomicity. However, since T_2 is committed and can't be aborted. Hence, a situation arrives where it is impossible to recover. And hence it is necessary that T_1 commits before T_2.

Recoverable schedules are desirable to prevent the system from going into inconsistent state permanently after failure occurs in the system.

4.7.2 Non Recoverable Schedule

- If any schedule S is not getting correct value of the data item after recovering it from failure is called as non-recoverable schedule.
- For example, the schedule S1 contains two transactions T_1 and T_2 as follows:
- Value of M is initially 100

T_1	T_2
Read(M);	
M=M+200;	
Write(M);	
	Read(M);
	M=M+100;
	Write(M);
	Commit;
Abort;	

- As shown in example Transaction T1 modifies the value of M to 300, then transaction T2 modifies value of M to 400 and T2 committed its transaction. After that, the transaction T1 aborted due to some failure in the system. Recovering it from failure leads to reassign the value of M to 400 (instead of 300) as it is committed value by T2 transaction. This type of scheduling is not recoverable.

4.7.3 Cascading Schedule

- Consider an example where three transactions are scheduled T1, T2 and T3:
- In which T2 reads the value of M which is written by T1 and T3 reads the value which is written by T2. Such schedule of transactions is known as cascading schedule.

T_1	T_2	T_3
Read(M);		
Read(N);		
M=5000;		
Write(M);		
Read(N);		
	Read(M);	
	M=10000;	
	Write(M);	
		Read(M);

- If a single transaction fails then series of transactions needs to be rollback. This rollback operation is called as Cascading rollback.

4.7.3.1 Cascaded Aborts

- When abort of Ti transactions leads to abort Tj transaction, this is called as **Cascaded aborts**.
- If any failure occurs in the T1 transaction then T1 needs to abort (abort operation simply rollbacks the performed operations so that it should recover from failure). As the T2 depends on T1 transaction, both needs to be aborted and rollback to recover from failure. Also T3 needs to abort and rollback because it is depend on the value provided by T2. This leads to cascading rollback/abort, this leads to unnecessary undoing the work.

4.7.4 Cascadless Schedule SPPU - May 14

University Question

Q. Explain cascade less schedules. **(May 2014, 4 Marks)**

- It is the types of recoverable schedule where, cascading rollback is not allowed. So that throughput of the system will increase.
- Consider an example where three transactions are scheduled T1, T2 and T3.
- In which T2 reads the value of M which is written by T1 and T3 reads the value which is written by T2. If any failure occurs in the T1 transaction, as the T2 is depend on T1 transaction, both needs to be rollback to recover from failure also T3 needs to rollback because it is depend on the value provided by T2. This leads to cascading rollback, this leads to unnecessary undoing the work.

T_1	T_2	T_3
Read(M);		
Read(N);		
M=5000;		
Write(M);		
Read(N);		
	Read(M);	
	M=10000;	
	Write(M);	
		Read(M);

- So it is necessary to avoid cascading rollbacks in schedules to increase throughput of the system.

The Cascadless schedule is a type of schedule where each transaction pair Ti and Tj is scheduling such that :

- Tj can read the data written by Ti.
- Commit operation of Ti must be performed before the commit operation of Tj.

If any failure occurs then no cascading rollback operations are needed. Only some parts of transactions need to be rollback.

4.7.5 Strict Schedule

It is the types of schedule in which no other transaction will perform Read(M) or Write(M) until the last transaction which performed on data item M is committed or aborted. Strict schedule is most restricted than Cascade less schedule.

T_1	T_2
Read(M);	
M=5000;	
Write(M);	
Commit;	
	Read(M);

Syllabus Topic : Concurrency Control - Need

4.8 Concurrency Control

- Concurrency control is an important concept of database management systems which is used to address conflicts with the concurrent access or modification of data that can occur with a multi-user system. This system helps to coordinate the multiple transactions executing simultaneously by preserving data integrity.
- Concurrency control ensures that correct results for concurrent operations are generated, while getting those results as quickly as possible.

4.8.1 Need of Concurrency Control

SPPU - Dec. 15

University Question

Q. Explain the need for concurrency control in transaction management. **(Dec. 2015, 5 Marks)**

Concurrency control is important in DBMS to achieve the simultaneous execution of the transactions. This can be done by executing few instructions of one transaction then next and the process is going on till the executions of transaction get completed.

Several problems can occur when concurrent transactions execute in an uncontrolled manner.

(1) The lost update problem

- Lost update problem occurs when two or more transactions are modifying the value of same data items simultaneously.
- In this the final value of the data item can be decided by the transaction which commits at last among those transactions.

Example,

- The initial value of both data items P and Q is 1000. The transaction T1 transfers 50 from P to Q. Transaction T2 withdraw 10% from amount of P.
- If T_1 and T_2 get executed serially then final values of P and Q are as follows:

If T_1 then T_2	If T_2 then T_1
Final values are P= 855 Q= 1050	Final values are P= 850 Q= 1050

- In above example T_1 lost its update due to the action of T_2.

(2) The temporary update (or dirty read) problem

The temporary update problem is known as Dirty Read Problem. It occurs when one transaction permitted to read the data item that is modified by an uncommitted transaction. The problem occurs when uncommitted transaction decides to rollback.

Example,

If transaction T_1 modify data item P and transaction T_2 is allowed to read the value of P as modified by T_1 before T1 commits. At this time dirty read problem occurs. The read operation which performed by the T_2 is called as dirty read operation.

(3) The incorrect summary problem

When one transaction performs aggregate function on several data items, and other transaction are updating those data. At that time incorrect summary problems occurs.

Example,

T_1 is calculating total balances of account x (100), account y (50), and account z (25). Meantime, T_2 has transferred 10 from x to z, so T_1 now has wrong result.

4.8.2 Different Concurrency Control Protocols

SPPU - May 15

University Question

Q. Explain different concurrency protocols in Database management systems.
(May 2015, 5 Marks)

To ensure transaction properties such as atomicity, isolation and serializability of concurrent transactions, concurrency control protocols are used. Concurrency control protocols can be broadly divided into two categories:

(a) Lock based protocols
(b) Time stamp based protocols

Syllabus Topic : Locking Methods

4.8.2.1 Locking Methods

A lock is a mechanism to control concurrent access to a data item. When multiple transactions are accessing some data item at a time, it may lead to generate some irrelevant data generation. Hence the data item should be isolated so that it should be accessible in a mutually exclusive manner that is when one transaction is accessing the data item, no other transaction should be able to access that item.

It can implement by a way in which a transaction can access the item only if it currently holds lock on the data item.

There are two types of locks.

(1) Shared Lock

The shared lock is also known as read lock. Shared lock is acquired on the transaction when concurrent transactions granted the permission for

read access on the data. A shared lock is issued when a transaction wants to read data from the database and no exclusive(write) lock is held on that data item.

(2) Exclusive Lock

This lock is also known as write lock. This lock is issued when a transaction needs to write or update a data item and there is no lock currently held on that item. This lock is placed on data item when a write or update operation is perform. Only one exclusive lock is placed on data at a time.

- In this system, every transaction will request for a appropriate lock (shared or exclusive) depending upon the type of operation which it want to perform on the database.
- There is a utility known as concurrency control manager which grants the locks to transactions. The transactions requests for a lock to this concurrency control manager.
- The locking mechanism allows reading the data items simultaneously for various transactions but allows only one transaction to perform write operation on the data item at a time.

4.8.2.2 Lock Based Protocols

A. Two-Phase Locking (2PL)

SPPU - May 13, Dec. 13, May 14, Dec. 14, May 16

University Questions

Q. Show that two phase locking protocol ensures conflict serializability. **(May 2013, 8 Marks)**

Q. Explain two phase locking protocol. How does it ensure serializability. **(Dec. 2013, May 2014, 8 Marks)**

Q. Write a short note on : Two phase locking protocol. **(Dec. 2014, 5 Marks)**

Q. Explain in brief two phase locking protocol. **(May 2016, 5 Marks)**

- In this scheme, each transaction makes lock and unlock request in 2 phases :
  1. **A Growing Phase(or An Expanding Phase or First Phase) :** In this phase, new locks on the desired data item can be acquired but none can be released.
  2. **A Shrinking Phase (or Second Phase) :** In this phase, existing locks can be released but no new locks can be acquired.
- In the beginning, the transaction is in growing phase in which it acquire lock as per requirement. After completing the work, the transaction releases the locks and enters into shrinking phase.
- The transaction cannot request for new locks after releasing the locks.

```
T
Lock-X(A);
Read(A);
A = A + 500;
Write(A);
Lock-X(B);
Read(B);
B = B - 100;
Write(B);
Unlock(A);
Unlock(B);
```

- The point in the schedule where the transaction acquires its final lock is called as **lock point**. This point is the end of growing phase. The transactions can be ordered as per their lock points. This sequence of transactions is the serailizability ordering for the transactions. This protocol assures serializability.
- In serializability the main issue is regarding write operation. The parallel write operations may create inconsistency in the database. The parallel read operations do not create any problem.
- In serial schedule, as the transactions are executed one after other, there is no risk of parallel write operations which may lead to inconsistency.
- The database transactions can be controlled in 2-Phase locking mechanism by applying the exclusive lock. The exclusive (write) lock is released in shrinking phase. Unless the exclusive lock of a data item is released by the transaction, other transaction can acquire exclusive lock on that data item. This helps to maintain the serailizability. It is not sure that Two-phase locking is free from deadlocks.

- In two-phase locking, there is possibility of cascading roll-back. This can be avoided by using **strict two-phase locking** in which a transaction must hold all its exclusive locks till it commits/aborts.
- **Rigorous two-phase locking** is more stricter. In it all the locks are held till commit/abort. Here transactions can be serialized in the order in which they commit.

B. Strict Two-Phase Locking

In Strict two phase locking protocol the first step of acquiring all the locks is same as first phase of two phase locking protocol. But the difference in Strict two phase locking and two phase locking is that Strict two phase locking protocol does not release the lock after using it. It holds all the locks till it does not reach to commit point. When it reached to the commit point it releases all locks at a time. Cascading rollback can be prevented by using the strict phase locking protocol. This protocol is used to make Cascadless schedules.

4.8.2.3 Timestamp Based Protocol

SPPU - May 13

University Question

Q. Explain Time Stamp Based Protocol. **(May 2013, 8 Marks)**

To identify the relative starting time of a transaction, DBMS creates a unique identifier called as Timestamp.

In general the values of Timestamp are assigned in the order in which the transactions are submitted to the system. Hence Timestamp is considered as a method of concurrency control in which transaction timestamp is assigned. To all the transactions.

Timestamps must have two properties namely :

- The timestamp should be unique.
- The timestamp values always increase.

- In timestamp method, as soon as a transaction is created, the timestamp it assigned to it. Timestamp for transaction is generated by using the system time. Timestamp is used to identify transaction uniquely.
- The older transaction has first priority to execute.
- The timestamp of transaction Ti is denoted as TS(Ti). Concurrency control techniques which are based on timestamp ordering doesn't use locks; so deadlocks cannot occur in the system.

(A) Timestamp Generation

- Timestamps can be generated in several ways.
- One way is to assign numbers to each transaction when it is created. For this, counter is used. The transaction timestamps are numbered as 1, 2, 3, ... n this scheme. The counter is reset to zero when no transactions are executing.
- System's current date/time value can be used to implement timestamps. It is another way to generate timestamp value.

(B) Timestamp Ordering Algorithm

- The timestamp-ordering protocol used to order the conflict read and write operation pairs in such a way that they maintain serializability. Timestamp ordering protocol ensures serializability of conflict read/write operation pairs.
- This protocol ensures serializability by assuring that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.
  - The timestamp of transaction Ti is denoted as TS (Ti).
  - Read timestamp of data item A is denoted by R-timestamp (A).
  - Write timestamp of data item A is denoted by W-timestamp (A).

Basic Timestamp ordering protocol works as follows

- Suppose that transaction Ti issue read(A) then either one of two conditions
  - If TS(Ti) < W-timestamp(A), then Ti reads overwritten value of A. So, the read operation on data item A is rejected, and transaction Ti is rolled back.
  - If TS(Ti) >= W-timestamp(A), then the read operation on data item A is executed, and R-timestamp(A) is set to the maximum of R-timestamp(A) and TS(Ti) i.e. R-timestamp(A)=max(R-Timestamp(A),TS(Ti))

- Suppose that transaction Ti issues Write(A)
  - If TS (Ti) < R-timestamp (A), then the system rejects the write operation and Ti is rolled back.
  - If TS(Ti) < W-timestamp(A), which means Ti is attempting to write an obsolete value of A. So, the write operation is rejected and Ti is rolled back.
  - Otherwise, the Write(A) operation is executed.

Thomas' Write Rule

This rule states :

- Suppose that transaction Ti issues Write(A).
  - If TS(Ti) < W-timestamp(A) and TS (Ti) < R-timestamp(A), then instead of rejecting the Write(A) operation it can be ignored and obsolete writes are deleted.

Syllabus Topic : Deadlocks

4.8.3 Deadlock SPPU Dec. 13

University Question

Q. When does Deadlock happen? How to prevent them and how to recover if deadlock takes place? (Dec. 2013, 8 Marks)

A system is said to be in a state of Deadlock, if every transaction in the schedule is waiting for another transaction in the schedule to release the lock of some data item. Consider transactions from T_0 to Tn. In this case transaction T_1 is waiting for T_2, T_2 is waiting for T_3 and so on. And at the last the transaction T_n is waiting for transaction T_0 in cyclic manner.

The transactions present in deadlock are either rollback or restarted. A system with a deadlock indicates bad behavior of the system. The rollback of the transaction may be partial. That is a transaction may rolled back to the point where it obtained a lock whose release resolves the deadlock.

It's not good to have any transaction that causes deadlock. To handle the deadlock situation, we have two options one is to avoid a system to enter in deadlock by Deadlock Prevention method or after deadlock try to recover it by deadlock detection and deadlock recovery method.

So to handle deadlocks following methods are used :

1. Deadlock Prevention
2. Deadlock Detection
3. Deadlock Recovery

4.8.3.1 Deadlock Prevention

- DBMS regularly checks all the transaction operations which are going to execute, to prevent any deadlock situation in the system. The checking of all the operations is done to analyze whether the execution of them leads the system in deadlock or not. If any operation of a transaction leads the system in deadlock state then that transaction will not allow for execution.
- There are different methods to prevent deadlock. All methods use the timestamp of transaction. The timestamp of any transaction Ti is denoted as TS(Ti).

Wait-Die Scheme

- In wait-die scheme if the older transaction requests resources hold by younger one then DBMS allows Ti to wait for needed resources but if the younger transaction requests resources hold by older one then DBMS kills Tj and restarted it later.
- Suppose if a transaction Ti requests for the resources which are already locked by other transaction Tj. Then in this scheme the DBMS checks for the timestamp of both the transaction Ti and Tj and performs following action.
- If TS(Ti) < TS(Tj) (means Ti is the older transaction than Tj) and Ti requests resources which are hold by Tj, then DBMS allows Ti to wait until resource is available for execution. That means if a younger transaction has locked some resource and older transaction needs those resources so older transaction is waiting for younger transaction to complete, then older transaction is allowed wait for it till it is available.
- TS(Ti) < TS(Tj) (means Ti is the older transaction than Tj) and Tj requests resources which are hold by Ti, then DBMS killed the Tj and restarted it latter with random delay but with the same timestamp. That means if the older transaction has

held some resource and younger transaction waits for the resource held by older one, then younger transaction is killed and restarted with same timestamp after some time.

Wound-Wait Scheme

- In this method, if an older transaction requests for a resource held by younger transaction, then older transaction forces younger transaction to kill the transaction and release that resource. The younger transaction is restarted later with same timestamp.
- But if the younger transaction is requesting a resource which is held by older one, then DBMS allows younger transaction to wait until the older releases that resource.

4.8.3.2 Deadlock Detection

Detecting deadlock situation in advance is always good instead of aborting a transaction. A deadlock avoidance mechanism is used to detect deadlock situation in advance. An algorithm that examines the state of the system is called periodically to check the occurance of deadlock.

In deadlock avoidance "wait-for graph" method is used.

Wait-for Graph

"Wait-for graph" scheme uses a graph(set of edges and vertices (nodes)) to detect deadlock in the system.

- A node is created when a transaction enters into system.
- When a transaction T_i requests for a lock on resource X which is held by some other transaction T_j, a directed edge is created from T_i to T_j.
- This edge is removed when T_j releases resource X, and T_i locks that resource.

Waits for graph indicates which transaction is waiting for another transaction to complete.

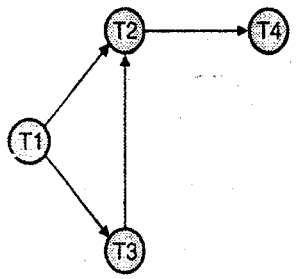


Fig. 4.8.1

A graph with cycle indicates the occurrence of deadlock in the system.

The wait for graph for Fig. 4.8.1 has following situation

- Transaction T1 is waiting for transaction T2 and T3
- Transaction T3 is waiting for transaction T2
- Transaction T2 is waiting for transaction T4

Here, the graph has no cycle; hence it is not in deadlock state.

Consider the transaction T4 is requesting an item held by T3 then a directed edge from T4 to T3 will be drawn.

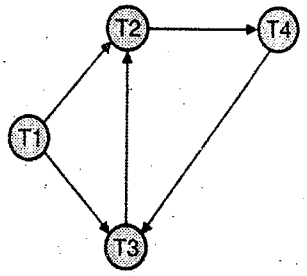


Fig. 4.8.2

Now the graph has a cycle – T2→ T4→ T3→T2

Here the transactions T2, T3 AND T4 are in deadlock state.

4.8.3.3 Deadlock Recovery

When deadlock is detected by the system, the only one solution is recovery of deadlock. The best way of recovery is rollback operation.

In the recovery process, three actions are taken

1. Selection of victim

The first step is to detect that which transactions should be rolled back to recover the system. It is necessary to rollback the transaction which has minimum cost. The minimum cost depends upon various factors :

- The data items used by the transaction.
- Time duration for the transaction to complete its assigned task.
- Number of total transactions need to be rolled back.

2. Rollback

After finalizing the transaction to be rolled back, we have to decide upto which previous point it should be rolled back.

Rollback operation has two options :

- **Total rollback :** Abort the whole transaction and restart it.
- **Partial rollback :**

  Rollback transaction as per necessity. But this format requires the system to store state of all the running transactions. The order of lock requests/grants and updates performed by the transaction should be recorded. The deadlock detection mechanism should decide which locks the selected transaction need to release to resolve the deadlock. The selected transaction should be released up to the point where the first lock is obtained. All the operations after that point should be undone.

The transaction must be capable of resuming execution after the partial rollback.

3. Starvation

In deadlock handling mechanism the selection of transaction to rollback(victim) depends upon the cost fact. Sometimes it is possible that same transaction may be selected multiple times in multiple deadlock situations. In such case it will be difficult for the victim transaction to complete its execution. It is called as **starvation**.

Hence it is necessary that, a transaction should not be selected as victim many number of times. To achieve this, the number of rollbacks of every transaction should be maintained in the cost factor.

Syllabus Topic : Recovery Methods

4.9 Recovery Methods

Failure is a routine situation for any system. There are various causes of failure.

1. **Transaction failure :** A transaction may fail mainly because of two reasons.
   a. **Logical error :** There are some internal conditions like invalid input, data not found, overflow or resource limit exceeded, because of which the system get failed and unable to continue its execution.
   b. **System error :** Because of cyclic dependency of transactions on each other , the system may enter in undesirable state like deadlock which leads to break the execution of transaction.
2. **System crash :** Because of hardware malfunction or bug in database software or operating system, the system may halt. The loss of contents of volatile storage is possible is this crash.
3. **Disk failure :** Failure during data transfer or head crash may cause he loss of data in a disk.

Whatever is the reason of failure, it leads to data loss and inconsistency in the database system, hence recovery is very important.

For DBMS recovery there are two types of techniques which maintain the atomicity of transaction are follows :

1. Log based Recovery
2. Shadow paging

Syllabus Topic : Log Based Recovery

4.9.1 Log Based Recovery

SPPU - Dec. 13, Dec. 15

University Questions

Q. Explain log based recovery scheme. **(Dec. 2013, 4 Marks)**

Q. Write a short note on : Log based Recovery. **(Dec. 2015, 5 Marks)**

Log

- Log is a data structure which is used to store all modifications performed on the database.
- Log contains sequence of log records which keeps track of update activities performed on the database.
- Log is an essential part of recovery system to handle the failures.
- So it is necessary to store the logs in storage where data should not be loss due to any reason. So the log records are stored on stable storage.

Log-based Recovery

- If any failure occurs in the system, with the help of logs, system can be recovered. This type of recovery is called as log based recovery.

- The Log-based recovery works in following manner
  - The log file is placed on a stable storage media.
  - When a transaction enters the system and starts execution, it writes a log about it.

<Tn, Start>

  - When the transaction updates an item X, it write logs as follows

<Tn, X, V1, V2>

  - It reads that the value of X is changed by Tn from V1 to V2.
  - When the transaction finishes, it logs

<Tn, commit>

- There are two techniques for using log to achieve recovery and ensure atomicity in case of failures.
- Deferred database modifications
  - This technique delay the database modification until transaction committed.
  - It allows storing all the modification done by transaction in the database, when that transaction committed successfully.
- Immediate database modification
  - Immediate database modification technique allows storing each modification done by transaction in to database as soon as it is done.
  - In this technique database modification operation doesn't wait for commit operation to be performed by transaction.
  - The system allows modifying the database when transaction processing is performed and not committed yet. Database modification performed by active transaction is known as uncommitted modification.

Syllabus Topic : Checkpoints

4.9.1.1 Checkpoints

- In DBMS storing large number of logs may fill out all the memory space available in the system. The size of log may increase tremendously as time goes which makes difficult to handle it. With the checkpoint mechanism we can remove all previous logs from the system and store them on a storage disc permanently. Using Checkpoint a point can be declared before which the DBMS was in consistent state and all the transactions were committed.
- In immediate database modification recovery technique, some transactions are requires to be redone and some requires to be undone by searching the entire log. So the searching entire log is time consuming process. To overcome this problem check points are used.
- Consider a system with concurrent transactions crashes and recovers, it behaves in the following manner

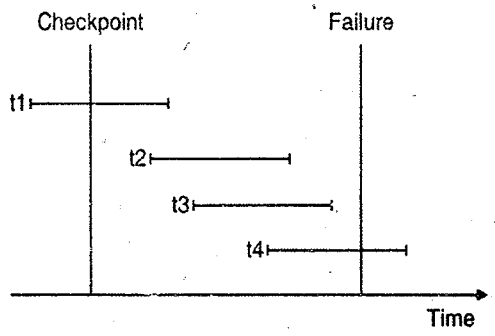


Fig. 4.9.1

- During execution along with keeping log using immediate database modification and deferred database modification technique, checkpoints are periodically performed by the system, which perform following sequence of steps :
  - All the logs reside currently on main memory are stored permanently on the disk.
  - All buffer blocks which are modified currently are written to the disks permanently.
  - A log record <checkpoint> also written onto the disk permanently.
  - While processing the checkpoint transactions are not allowed.
  - One advantage of maintaining checkpoint is that, if in set of transactions <Ti , ... , Tn> the most recent checkpoint is maintained at transaction Tj where, i<j and j<n. i.e. Tj is in between Ti and Tn.
  - Then for recovery purpose keeping log records of transactions which performed in between Tj to Tn is preferable. And deleting

the unnecessary log records which are done before the checkpoint Tj is allowed.

- This means that the transaction recovery can start from the checkpoint log record and not from first record in the log. So it reduces the time.

Syllabus Topic : Shadow Paging

4.9.2 Shadow Paging SPPU - Dec. 13

University Question

Q. Explain shadow paging recovery scheme. **(Dec. 2013, 4 Marks)**

- The shadow paging does not require the use of a log in a single-user environment. In this scheme database is considered to be made up of a number of fixed-size disk pages or blocks for recovery purposes.
- A directory with n entries is created. In it the i^{th} entry points to the i^{th} database page on disk. This directory is kept in main memory. It records references of all reads or writes to database pages. When execution of transaction started we copy the current directory whose entries points to the most recent database pages on disk into a shadow directory.
- The current directory is used by the transaction and the shadow directory is saved on disk.
- While executing the transaction, no changes are made in shadow directory. If a 'write' operation is performed, then new copy of modified database page is created. Both new and old copies are kept. That means for pages which are updated during the transaction, old and new, both versions are kept. The shadow directory refers the old version while the current directory refers the new version.
- In case of failure, for recovery purpose the modified database pages are freed and current directory is discarded. The shadow directory provides the state of the database before transaction execution. The state is recovered by restoring the shadow directory. In this way the database returned to its state prior to the transaction that was executing when the crash occurred and any modified pages are discarded.
- This technique can be categorized as a NO-UNDO/NO-REDO technique for recovery of database because for recovery, neither undoing nor redoing data items is performed.
- In case of multiuser system where transactions are performed concurrently, logs and checkpoints are used in the shadow paging technique.

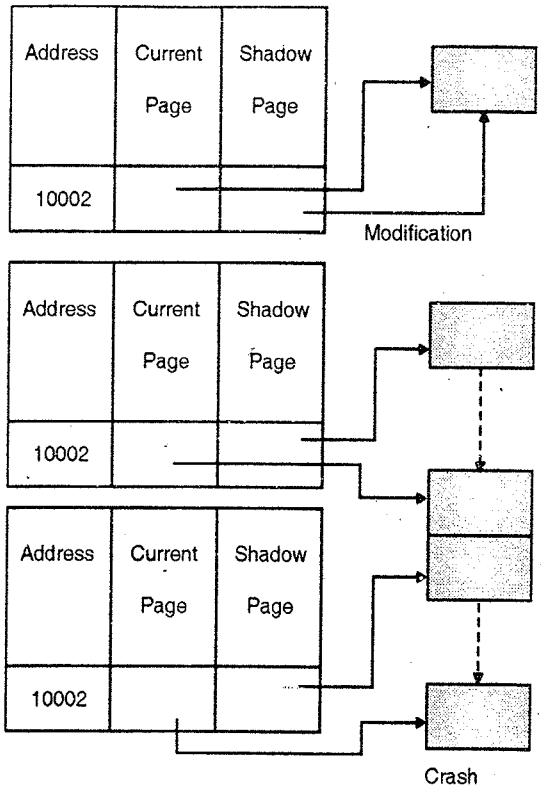


Fig. 4.9.2

- One of disadvantage of this technique is that the database pages which are updated, change location on disk. Because of this, it is difficult to keep related database pages close together on disk. It requires complex storage management strategies to manage those pages.

Syllabus Topic : Query Processing

4.10 Query Processing

The database is used to store the data at one place. This data is accessed and manipulated by the user. The process of accessing and manipulating the data should be done efficiently i.e. it should be accessed easily and quickly. To access the data user has to communicate with database. User requests for the data in language like SQL. SQL is a high level language but the underlying systems in the DBMS do not understand

SQL. There has to be some low level language to which these systems can understand. Hence request query is accepted in SQL and converted into low level language. This process is known as query processing.

Query processing defined as the set of activities performed to convert SQL high level queries into low level languages and then retrieve data from database by using this low level language. The set of activities are known as the steps involved in query processing which are explained below.

4.10.1 Basic Steps in Query Processing

SPPU - Dec. 13, May 14

University Question

Q. What are the steps involved in query processing? Explain each in brief.

(Dec. 2013, May 2014, 9 Marks)

Steps of query processing are follows

1. Parsing and translation
2. Optimization
3. Evaluation

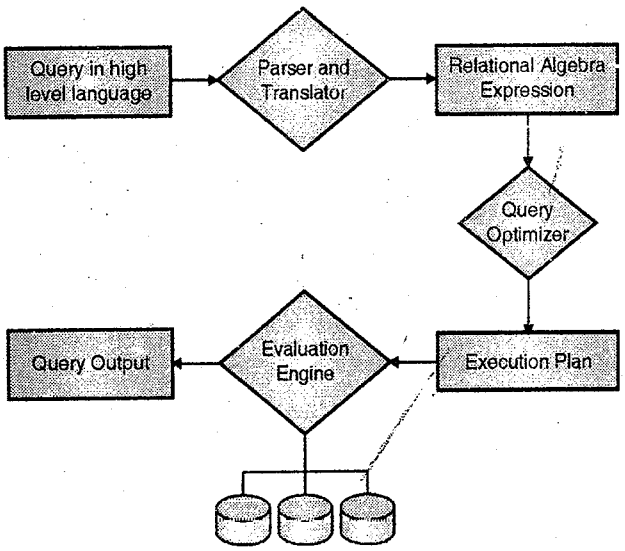


Fig. 4.10.1 : Steps of query processing

1. Parsing and Translating the Query

- The query is by default in high level format. In processing a query submitted to a DBMS, the first step is to convert the query into a form usable by the query processing engine. In high level languages, the query is represented as a string (sequence of characters).
- These sequences of characters may represent various types of tokens like keywords, operators, operands, literal strings, etc. The parser extracts the tokens from the sequence of characters and translates them into the related internal data elements (i.e. relational algebra operations and operands) and structures (i.e. query tree, query graph). Verifying the validity and syntax of the original query string is the main job of parser.

2. Optimizing the Query

- This is the second stage of query processing in which the query processor applies rules to the internal data structures of the query to transform them into equivalent representations which are more efficient.
- These rules are based on mathematical models of the relational algebra expression. The main function of the query optimization engine is to select correct rules to apply, situation to apply these rules and format to apply them.

3. Evaluating the Query

- This is the last step in query processing in which best evaluation plan created by the optimization engine is selected and then performed.
- There are various methods to execute the query. A whole query can be executed or small parts of query can be executed simultaneously. The important thing is that the results of all the methods should be same.

Syllabus Topic : Query Optimization

4.11 Query Optimization

- Query optimization is a function of many relational database management systems. The query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans.
- Generally, the query optimizer cannot be accessed directly by users: once queries are submitted to database server and parsed by the parser, they are then passed to the query optimizer where optimization occurs.

- A query is a request for information from a database. It can be as simple as "finding the address of a person with SS# 123-45-6789," or more complex like "finding the average salary of all the employed married men in California between the ages 30 to 39, that earn less than their wives."
- Queries results are generated by accessing relevant database data and manipulating it in a way that yields the requested information.
- Since database structures are complex, in most cases and especially for not-very-simple queries, the needed data for a query can be collected from a database by accessing it in different ways, through different data-structures, and in different orders.
- Each different way typically requires different processing time. Processing times of the same query may have large variance, from a fraction of a second to hours, depending on the way selected.
- The purpose of query optimization, which is an automated process, is to find the way to process a given query in minimum time.
- The query optimization typically tries to approximate the optimum by comparing several common-sense alternatives to provide in a reasonable time a "good enough" plan which typically does not deviate much from the best possible result.

Implementation

- The query plan is selected by query optimizer. Most query optimizers represent query plans as a tree of "plan nodes".
- A plan node encapsulates a single operation that is required to execute the query. The nodes are arranged as a tree, in which intermediate results flow from the bottom of the tree to the top.
- Each node has zero or more child nodes those are nodes whose output is fed as input to the parent node. For example, a join node will have two child nodes, which represent the two join operands, whereas a sort node would have a single child node (the input to be sorted).
- The leaves of the tree are nodes which produce results by scanning the disk, for example by performing an index scan or a sequential scan.

Join ordering

- The performance of a query plan can be determined mainly by the order in which the tables are joined.
- For example, While joining 3 tables X, Y, Z of size 10 rows 10,000 rows, and 1,000,000 rows, respectively, a query plan which is used to join Y and Z first can take several orders-of-magnitude more time to execute than one that joins X and Z first. Number of query optimizers uses dynamic programming algorithm to determine join order.

Syllabus Topic : Performance Tuning

4.12 Performance Tuning

- **System Performance** depends on how fast the DBMS queries are executed. If the query execution speed is faster, then the system performance also increases.
- If a system is slow or unresponsive we can say that the performance of that system is low. Performance problem usually occurs because of high loading on the system.
- Performance tuning done for the improvement of system performance by removing the problems related to performance. Performance problems occur in the computer system can be either real or anticipated. A synonym of performance tuning is the scalability of computer system. Scalability is defined as the ability of computer system to handle higher loads. Most systems may handle increased load but performance decreases.
- Following steps are followed by Systematic tuning :
  1. Before any updation done in the system, performance of the system is measured.
  2. Establish numeric values by assessing the problem. Those values are used to categorize acceptable behavior on the system.
  3. Identify bottleneck, Bottleneck is nothing but the part of the system due to which it is critical to improve the performance.
  4. Remove the bottleneck by modifying the part of the system which becomes a bottleneck to the improving performance.

5. After updation measure the performance of the system.
6. If the updation results into better performance, then adopt it. Otherwise, put it back the way it was.

- These steps are followed in a cyclic manner until system provides expected performance.

Example,

Consider the client-server performance tuning :

- At client side, performance of SQL is tuned by generating queries that executed using minimum resources and produces correct results in least amount of time.
- At server side, performance of DBMS is tuned by configuring the DBMS environment in such a way that response time for client's requests will be as possible as less. And also to respond those requests it uses existing resources in efficient manner.

❑❑❑

UNIT V

CHAPTER 5

Parallel and Distributed Databases

Syllabus

- Introduction to Database architectures: Multi-user DBMS architectures
- Case Study- Oracle Architecture
- **Parallel Databases :** Speedup and Scaleup, Architectures of Parallel Databases.
- **Distributed Databases :** Architecture of Distributed Databases, Distributed Database Design, Distributed Data Storage
- Distributed Transaction : Basics, Failure modes, Commit Protocols, Concurrency Control in Distributed Database

Syllabus Topic : Introduction To Database Architecture - Multi-user DBMS Architectures

5.1 Introduction to Database Architecture : Multi-user DBMS Architectures

The common architectures that are used to implement multi-user database management systems :

1. Teleprocessing
2. File-Server
3. Client-Server

5.1.1 Teleprocessing

- Teleprocessing is a traditional architecture for multiuser systems. This architecture has one computer with single Central Processing Unit(CPU) and multiple terminals. Here the processing is performed in one physical computer.
- The different terminal are typically "dumb", incapable of functioning on their own and cabled to the central computer.
- In this system, the terminals can access DBMS and database directly but this it also makes very heavy load to the CPU as it has to perform two operations
  - Run application programs and DBMS.
  - Formatting of data to represent on the terminals.

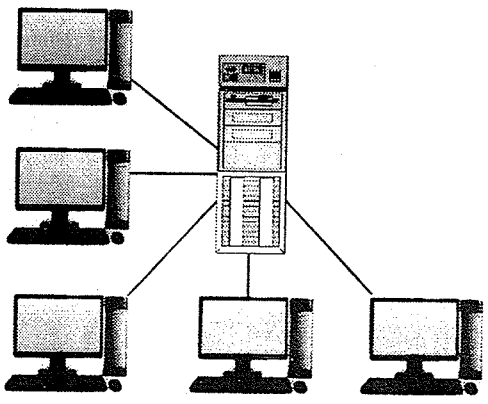


Fig. 5.1.1 : Teleprocessing architecture

5.1.2 File Server

- In the file-server architecture, there is a computer which is connected to a network and mainly serves as a shared storage.

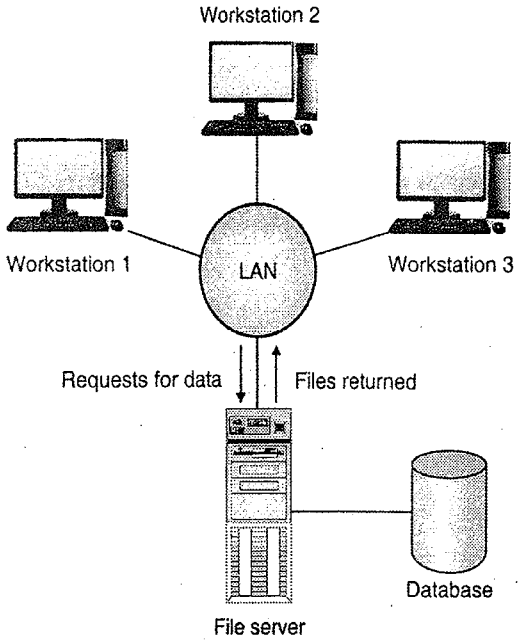


Fig. 5.1.2 : File server architecture

- The processing is distributed over the network in the file-server architecture is typically the Local Area Network (LAN). The file server stores all the files which are required to applications and DBMS. To get these files, the applications and DBMS has to make requests to file server.
- The file server works as shared data disc.

Disadvantages

- It generates heavy network traffic.
- Each workstation requires a full copy of DBMS.
- Complex integrity, concurrency, and recovery control : Multiple DBMSs may concurrently access the same shared file.

5.1.3 Client Server Database Architecture

SPPU - May 14, Dec.-15

University Questions

Q. Explain client server database architecture. **(May 2014, 3 Marks)**

Q. Explain Client Server Architecture with suitable database application. **(Dec. 2015, 5 Marks)**

- A network architecture in which each computer or process is connected in the network is client or server is known as **client server architecture**.
- Server is nothing but the powerful computer. It has more storage and processing capacity than the client machine. It manages client computers, printers, disk drivers and network traffic.
- Clients are the PC's where user can run the applications. Clients always depend on server for resources and processing power.
- In this architecture clients request data from the server and server provide all the required information to the client in response.

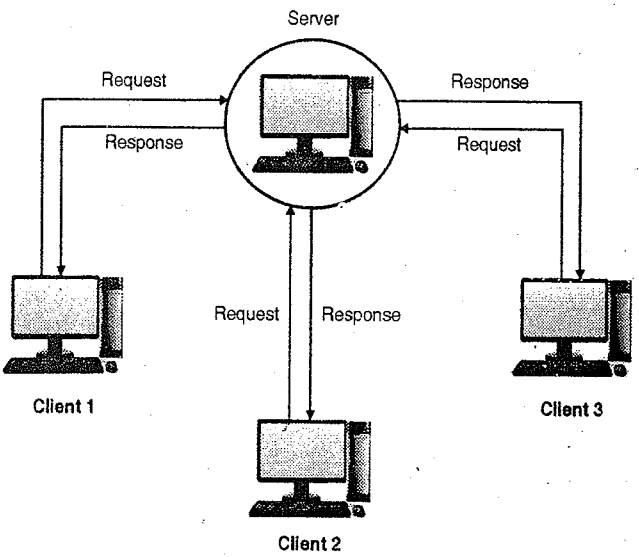


Fig. 5.1.3 : Client-server architecture

Banking Database Application of Client Server Architecture

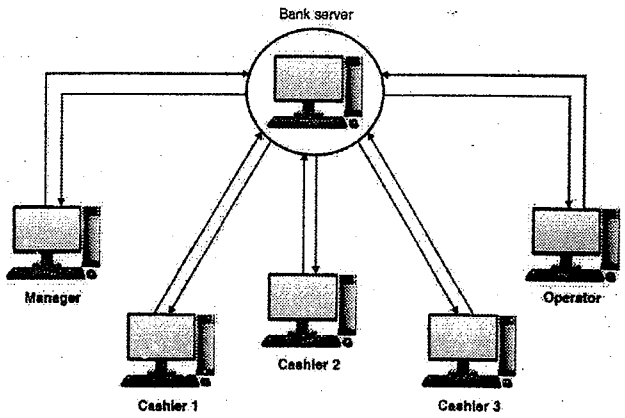


Fig. 5.1.4 : Bank database system

- Consider the Banking Database Example. In this system, the database is stored on server. From this server all clients like Manager, Operator and Cashiers in the system access the data from database. As per roles and privileges granted the data accessibility rights are decided for different clients in the bank.
- For example, the manager will have access right to whole data while the operators and cashiers can access data as per their tasks assigned.

- In the banks like SBI and ICICI where the number of customers or account holders is more, it is not sifficient to keep only one cashier to reciept cash and one cashier to pay cash. For this tasks there are multiple cashiers.
- Consider, account holder 'A' wants to withdraw some amount from his bank account. In this case as shown in Fig. 5.1.4, there are number of cashiers and 'A' can goto any one of them. This is possible because the data of all the account holders is available on the server. And all the cashiers can access the data. This definitely increases the efficieny of bank.

Advantages of Client Server Architecture

- Organizations always try to maintain service and quality competition to sustain its market position with the help of advanced technology where the client/server model makes an effective impact.
- Implementation of client/server architecture in an organization will definitely increase productivity through the usage of cost-effective user interfaces, enhanced data storage, strong connectivity and reliable application services.
- There are number of advantages of Client Server Architecture
  1. **Centralized Database :** The database is centrly available for all the clients easily. The centralized database is easy to manage for database administrator.
  2. **Security :** Rules defining security and access rights can be defined at the time of set-up of server.
  3. **Back-up and Recovery possible :** As all the data is stored on server, it is easy to take periodical backup of it. Also, in case of any break-down if data is lost, it can be recovered easily and efficiently.
  4. **Upgradation and Scalability :** Making Changes can is easy by just upgrading the server. The new resources and systems can be added by making required changes in server.
  5. **Server Role :** Server can play different roles for different clients.

5.1.3.1 Types of Client Server Database Architecture

Now to understand types of Client Server database architecture first we have to study the concept of layers or services.

Layers or Services

A software application is created using programming languages(called as frontend) and database(called as backend). In every software we have to implement following three layers.

1. **User Layer(presentation layer)**
   - It is also called as client layer which contain User interface of our application. This layer is used for design purpose. In this data is presented to the user and also input can accepted from the user.
   - For example in banking software, the registration form of an account holder can be considerd as user layer.
2. **Business Layer**
   - This layer is also known as business layer. In this layer we can write all business logic like validation of data, calculations, data insertion etc. This acts as an interface between Client layer and Data Access Layer.
   - This layer is also called the intermediary layer helps to make communication faster between client and data layer.
3. **Data Layer**
   - In this layer actual database is comes in the picture. Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.
   - Depending upon the implementation of these three layers there are types of database architecture.

A. Two tier architecture
B. Three tier architecture

A. Two Tier Architecture

SPPU - Dec. 13

University Question

Q. Explain two-tier architecture.

(Dec. 2013, 2 Marks)

- The **two tier architecture** is based on the client server architecture. The direct communication takes place between client and server.
- The two tier architecture is the architecture in which user interface is run on client side and data layer is stored on the server side. In two tier architecture we can integrate business layer with either presentation layer or database layer or can be distributed in both.

i) The business layer can be integrated with presentation layer at client side. In this case the size of client application increases, hence it is known as **Fat Client.**

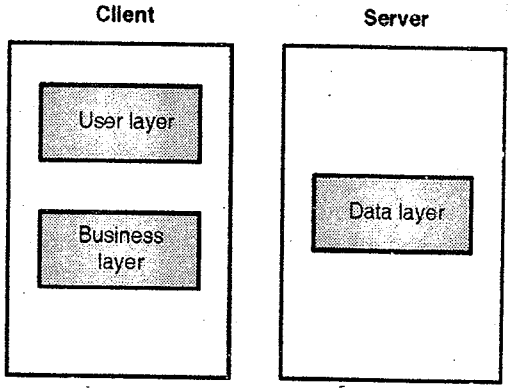


Fig. 5.1.5 : Fat client

ii) The business layer can be integrated with data layer at server side. In this case the size of server application increases, hence it is known as **Fat Server.**

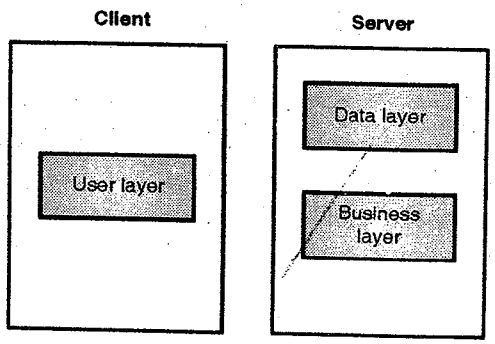


Fig. 5.1.6 : Fat server

iii) The business layer can be integrated with both user layer and data layer.

Advantages of two tier architecture

1. In two tier architecture, applications can be easily developed due to simplicity.
2. In this client and server are directly connected, due to which communication becomes faster.
3. Maximum user satisfaction is achieved with accurate and fast prototyping of applications through robust tools.
4. It contains static business rules which are easily applicable for homogeneous environment.
5. We can integrate application layer physically with the database layer as well as user interface layer

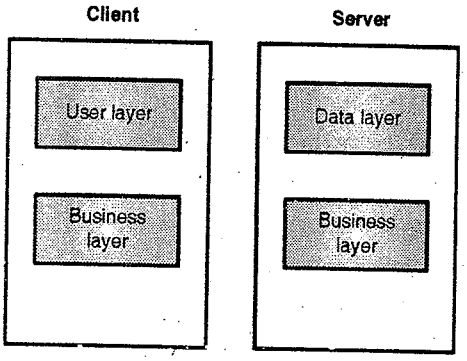


Fig. 5.1.7 : Two tier architecture

Disadvantages of two tier Architecture

1. It can only support to the limited number of users due to lack of scalability.
2. The performance of two tier architecture degrades when number of user increases.
3. Two tier architecture is cost ineffective.
4. As per security concern it is complicated.

B. Three Tier Architecture SPPU - Dec. 13

University Question

Q. Explain three-tier architecture. **(Dec. 2013, 2 Marks)**

- **The three tier architecture** is most widely used architecture in today's world.
- In this architecture the user layer, business layer and data layer are implemented independently by three different applications.
- The data required by the business logic exists in database server.
- In three tier architecture all layers interact with each other independently.

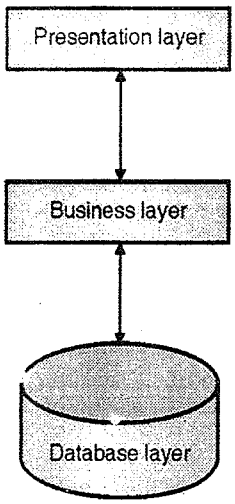


Fig. 5.1.8 : Tree tier architecture

Advantages of Three tier Architecture

1. In three tier architecture we can manage the data independently.
2. We can make the changes in presentation layer without affecting other two tiers.
3. As each tier is independent it is possible to use different groups of developers
4. It is most secure since the client doesn't have direct access to the database layer.
5. When one tier fails there is no data loss, because you are always secure by accessing the other tier.
6. Due to distributed deployment of application server, scalability is increased.
7. A similar logic can be used in various applications. It is reusable.
8. It is robust and secure due to multiple layers.

Disadvantages of three tier architecture

1. It is more complex structure.
2. More difficult to set up and maintain it.
3. The physical separation of the tiers may affect the performance.

Example : SPPU - May 15, May 16

University Question

Q. Explain 3-tier web architecture with diagram for online shopping database system?
(May 2015, May 2016, 8 Marks)

This is the online shopping diagram for 3 Tier architecture. Here the as a frontend Dot Net environment is used while as backend database MS SQL Server 2008 is used.

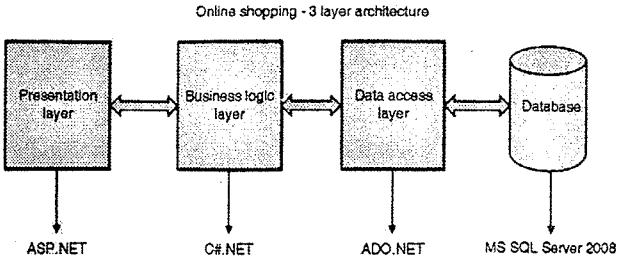


Fig. 5.1.9 : Online shopping diagram for 3-tier architecture

Fig. 5.1.9 shows three layers :

1. **Presentation layer :** This layer consist of user interface designed for the interaction with end user. This layer is created in ASP .Net. It includes the screens which will be used by the end user for shopping. Theses screen show the products with details as per their categories. User can select the product to purchase and add them into cart. This design is created with advanced controls available in ASP .Net.
2. **Business Layer** : This layer consist of validation checking code related to product selection of user. Accidently user may select wrong number of products to purchase. For example, if any user is giving order to purchase 10000 TV sets, then the order should be validate. This logical code is implemented using the C# .Net.
3. **Data Layer** : This layer contains the code interacting with database on the server. For example, accessing product details from database, inserting transaction details of user order in database etc. This database handling is implemented using the ADO .Net. Here all the three layers work independently and efficiently.

5.2 Centralized Database Architecture

SPPU - Dec. 13, May 14

University Questions

Q. Write short note on centralized database system. **(Dec. 2013, 3 Marks)**

Q. Explain centralized database architecture. **(May 2014, 3 Marks)**

A database which is located, stored and

maintained at single location is known as **centralized database architecture**. This location is most often a central computer or database system, for example a desktop or server CPU, or a mainframe computer.

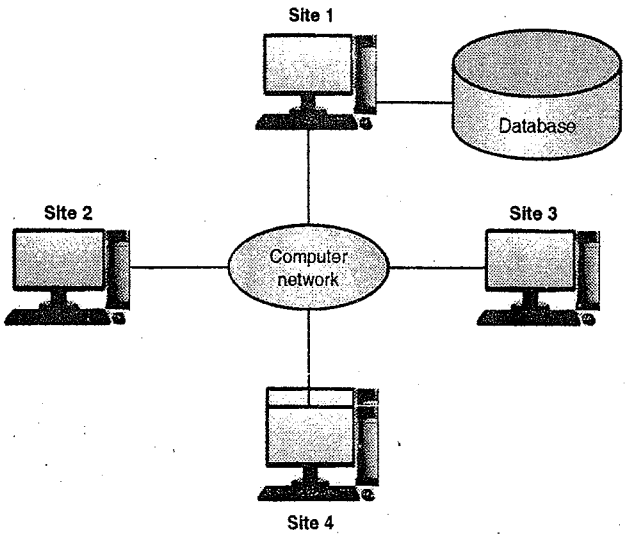


Fig. 5.2.1 : Centralized database architecture

In centralized database architecture the database physically resides on only one site and other sites can access it through the network.

Advantages of centralized database system

1. Easy to use by end-users due to its simplicity provided by storing all the data at one place. It helps to maximize Data integrity and also minimized data redundancy.
2. Data reliability is enhanced and accuracy and consistency of data is maintained.
3. It provides better security for stored data. As all the data of any organization is stored on one place, the organization can easily focus on security detail of one place rather than of multiple locations.
4. In a centralized system, information can be changed or updated easily.
5. Centralized system provides fault tolerance facility so the data is preserved in better way.
6. Also it is cost effective approach as this system requires minimum maintenance cost.

Disadvantages of centralized database system

1. It is highly dependent on network connectivity, hence if the network is slower, then the time require to access data from database is also increases.
2. It decreases the efficiency of the system, as there is only one copy of data. If more than one users want to access it then it is not possible.

Syllabus Topic : Case Study – Oracle Architecture

5.3 Case Study – Oracle Architecture

Oracle Database

Oracle database is made up of two components :

1. Instance 2. Database

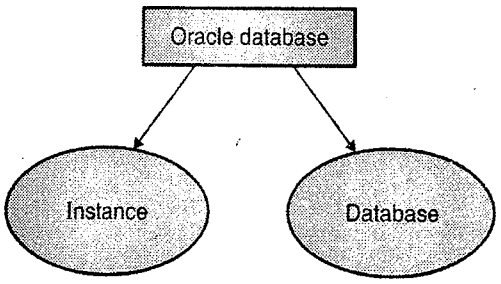


Fig. 5.3.1 : Components of oracle database

1. Instance

In database files the database structure and processes are very important. An instance is nothing but the memory structure and processes that are used to access the data from the database.

The **memory structure** is consist of

- System Global Area (SGA)
- Program Global Area (PGA)
- Optional Software Code.

The background processes are

- Database Writer (DBWn)
- Log Writer (LGWR)
- Checkpoint (CKPT)
- System Monitor(SMON)
- Process Monitor(PMON)
- Optional – Archiver (ARCn), Recoverer (RECO)

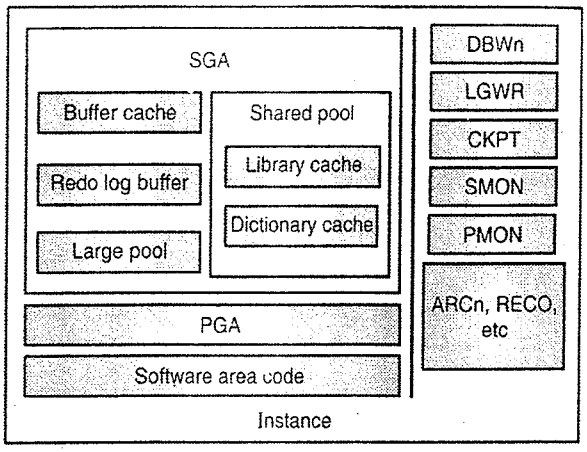


Fig. 5.3.2

Now we will discuss all these components in detail :

- **System Global Area (SGA)**

  This is the primary part of oracle structure. The SGA is a memory area for structures that are shared among users. The main components of SGA are Buffer Cache, Shared Pool, Redo Log Buffer, Large Pool and Java Pool.

  ### Buffer Cache

  This cache is used to store the frequently accessed data blocks from tables and indices in memory. It reduces the need to perform physical disc IO. The size of this buffer cache can be changes as per the requirement.

  ### Shared Pool

  - Oracle is designed for multiuser systems. When multiple users executes sme SQL query then they can share the data structures that represent the execution plan for these statements. For this purpose the data the data which is local to each specific call of the statement should be kept in private memory.
  - The Shared pool is used to store the sharable parts of data structures representing the SQL statement with text of the statement. This helps in reducing the compilation time since new call of a statement which is already cached does not have to go through the complete compilation process.
  - The **library cache** is used to store the information about the commonly used SQL statements while the **dictionary cache** is used to store the information about object definition like table, columns, indexes, privileges etc.
  - **Redo Log Buffer :** The DML statements like select, insert, update or delete generates redo entry. This redo entry is nothing but all the information about changes made by user. To store this redo entry, redo log buffer is used before it is written in to redo log files.
  - **Large Pool :** This is basically optional area in the SGA. It is used for I/O processes and it also helps to relieve the burden of shared pool.
  - **Program Global Area (PGA) :** When the SQL statement is parsed, its result is stored in library cache. The value of binding variable is stored in the PGA to make it private so that it should not be accessed by other users.
  - **Software Area Code :** In software area code, the oracle software application resides.

- **Dedicated Server : Process Structure**

  To execute Oracle server code, there are two types of processes : **Server processes** which processes the SQL statements and **background processes** that performs various performance and administrative related tasks. Some of these processes are as follows.

  - **Database Write(DBWn) :** Before removing from the buffer cache, the Database Writer writes the buffer into the disc if it has been modified. It improves the performance of the system.
  - **Log Writer (LGWR) :** The log Writer writes the redo entries from redo log buffer into the redo log files.
  - **Checkpoint (CKPT) :** The checkpoint process updates the headers of the data file when a checkpoint occurs.
  - **System Monitor (CMON) :** This process is used for crash recovery when any process fails. It also manages the space by reclaiming the unutilized space in temporary segments.
  - **Process Monitor(PMON) :** This process is used for process recovery when any process fails. It releases the resources and performs various cleaning operations.
  - **Archiver (ARCn) :** When the online log files fills up, the Archiver copies the online redo log file to an archived redo log.

- **Recoverer (RECO) :** This process resolves failures and conducts cleanups for distributed transactions.

2. Database

The database refers to disc resources. It is basically divided into two types : Logical and Physical structure.

A) Logical Structure : To manage the data efficiently, the oracle database is divided into smaller units like tablespace, segment, extent and data block.

- **TableSpace :** Tablespace is the collection of logical database objects.

  There are three types of tablespaces

  (a) Permanent (b) Undo

  (c) Temporary

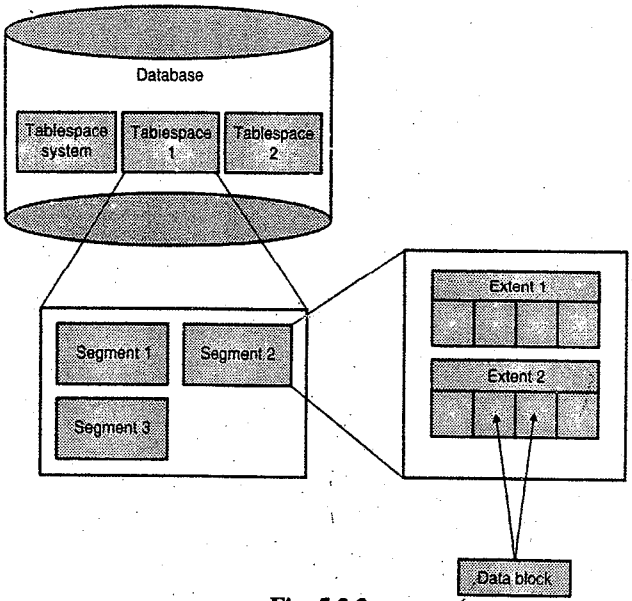


Fig. 5.3.3

- **Segment :** The tablespace is divided into segments. Same type of objects are stored in the segments. There are following types of segments in oracle :

  Table, Index, Cluster, Rollback, Temporary, Cache etc.

- **Extent :** A segment is then divided into extents. It consists of data blocks. An extent is allocated for the enlarged database object.

- **Data Block :** It is the smallest unit of storage in the oracle database.

B) Physical Structure : It consists of data files, redo log files and control files.

- **Data Files :** Data files corresponds to tablespace

- **Redo Log Files :** When a transaction is committed the details about the transaction in the redo log buffer is written into redo log file. This file helps in recovery when failure occurs.

- **Control Files** : The information about the physical structure of database is stored in the control file.

Syllabus Topic : Parallel Database

5.4 Parallel Database

Now a day organizations need to handle huge amount of data with high transfer rate. For such requirement the client server or centralized system is not efficient. The need to improve the efficiency of system, the concept of Parallel Databases comes in picture.

Parallel database improves the performance of processing of data using multiple resources simultaneously. Multiple CPU, Disks can be used simultaneously. A parallel database improves speed of data processing. A parallel server can allow access to a single database by users on multiple machines, with increased performance. By doing parallelization of loading data, building indexes and evaluating queries, the parallel database system improves the performance. In parallel database system we can use thousands of small processors.

Advantages of Parallel Database

1. **Performance Improvement :** By connecting multiple resources like CPU and disks in parallel we can significantly improve the performance of system.

2. **High Availability :** Same data can be stored on multiple locations so that the availability of data can be increased. In parallel database nodes has less contact with each other, so failure at one node does not cause for failure of entire system. One of the surviving nodes recovers the failed node and the system continues to provide data access to users. This means data is much more available than it would be with a single node upon node failure. This also amounts to significantly higher database availability.

3. **It increases Reliability :** When a site fails, the execution can continue with another available site which is having copy of the data. Due to this it becomes more reliable.
4. **It have large capacity :** In parallel database more users request access to the database due to which administrator add more computers to the parallel server. The addition of computers boosts the overall capacity.

Syllabus Topic : Speedup and Scaleup

5.4.1 Speedup and Scaleup

SPPU - May 15, May 16, Dec. 16

University Questions

Q. Explain speedup and scaleup in parallel databases in detail.
(May 2015, May 2016, 5 Marks)

Q. What is speedup and scaleup attributes in parallel database architectures?
(Dec. 2016, 4 Marks)

To measure the performance of parallel processing we can use two important properties:

1. Speedup

- The extent in which more hardware can perform the same task in less time than the original system is known as **Speedup.** It means the execution of task done in less time by increasing degree of parallelism. The time which is required to process a task is inversely proportional to the time when number of resources are used.
- With good speedup, additional processors reduce system response time. We can measure speedup is as follows:

$$\text{Speedup} = \frac{\text{time\_original}}{\text{time\_parallel}}$$

Where,

Time_Parallel is the time spent by a larger, parallel system on the given task

- For example, If the original system took 60 seconds to perform a task, and two parallel systems took 30 seconds, then the value of speedup would be 60/30 = 2.

Linear Speedup

We can say that speedup is linear when speedup is equal to N. That is the elapsed time of small system is N times larger than the elapsed time of large system where N is the number of resources.

Sub-linear Speedup

When speedup is less than N then it is called as sub-linear speedup.

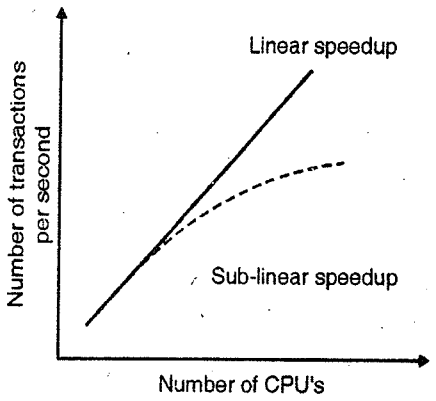


Fig. 5.4.1 : Speedup

2. Scaleup

- The ability to keep the same performance level when both workload and resources increase proportionally is known as **Scaleup.** It is the process of handling large task in same amount of time by increasing degree of parallelism.

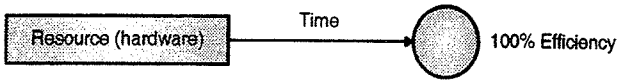
  We can measure the ScaleUp is as follows:

$$\text{Speedup} = \frac{\text{Volume\_parallel}}{\text{Volume\_original}}$$

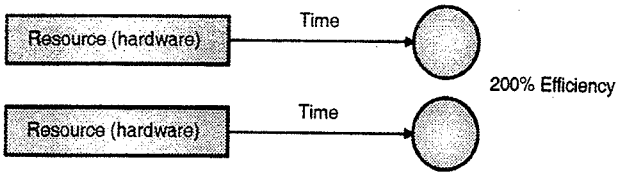
  Where Volume_Parallel is the transaction volume processed in a given amount of time on parallel system.

- In case of good Scaleup if there is increase in transaction volume, then to keep response time constant, we can add hardware resources like CPU.

Original System



Parallel System



5.4.1.1 Different Factors Affecting the Speedup and Scaleup Attributes

SPPU - Dec. 16

University Question

Q. Explain the different factors affecting the speedup and scale-up attributes. (Dec. 2016, 4 Marks)

Startup costs

For all the parallel operations, there is a associated cost involved in starting the process. When a big process is break in small processes then it consumes some amount of time and recourses. If we want to execute a query in parallel then we need to partition the table and instructs various processors to execute the query in parallel.

Consider a query to sort the data.

```
Select * from emp order by ename
```

Now to execute this query in parallel we need to partition the table as per sorting criteria and instructs various processors to execute the query in parallel. Both of these operations need some amount of time before the parallel execution.

- **Interference :** The various resources are shared by all he processes which executes in parallel. Whenever interference of a new process happens, it leads to slow down the overall access of all the processes. This affects both speed-up and scale-up.
- **Skew :** It is difficult to break a big tasks in equal size small tasks. In such case the performance of the system depends upon the slowest CPU which processes the largest sub task. This type of uneven division of big tasks into smaller ones is called as skew. This also affects the speed-up and scale-up.

Syllabus Topic : Architecture of Parallel Databases

5.4.2 Architecture of Parallel Database

SPPU - Dec. 14, May 16

University Questions

Q. Explain any two parallel database architectures in details. (Dec. 2014, 5 Marks)

Q. Explain parallel database architectures. (May 2016, 4 Marks)

There are four different architectural models of parallel database:

1. Shared memory
2. Shared disk
3. Shared nothing
4. Hierarchical

5.4.2.1 Shared Memory

SPPU - Dec. 15

University Question

Q. Explain Shared Memory Parallel Database System architecture. (Dec. 2015, 3 Marks)

In this architecture of parallel database common memory is shared among the multiple processors. These processors are connected through the interconnection network to the main memory and disk. The connection used in this architecture is usually high speed network connection which makes data sharing easy. Shared memory architecture have large amount of cache memory at each processors.

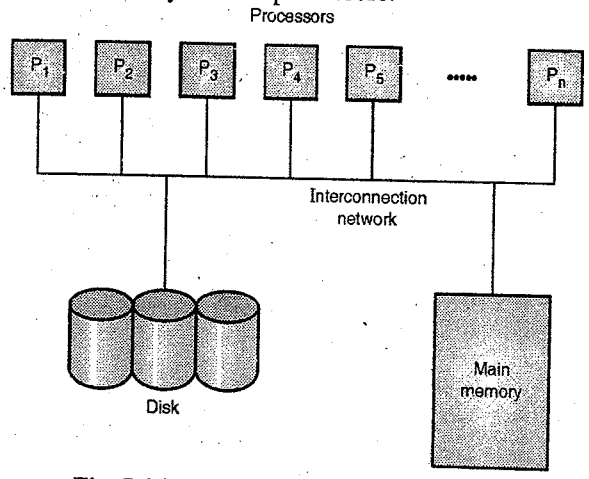


Fig. 5.4.2 : Shared memory architecture

Advantages of shared memory architecture

1. Any processor can access data easily.
2. Effective communication between processors through common memory address space.
3. Communication overheads are less.

Disadvantages of shared memory architecture

1. Waiting time of processors is increased due to more number of processors
2. Degree of parallelism is limited.
3. Addition of processor slows down the existing processors.
4. When a processor tries to access the data updated by other processors, then we have to take care that the data is of latest updated version.

5.4.2.2 Shared Disk Architecture

In the shared disk architecture of parallel database system single disk is shared among all the processors. These all processors also have their own private memory which makes data sharing efficient.

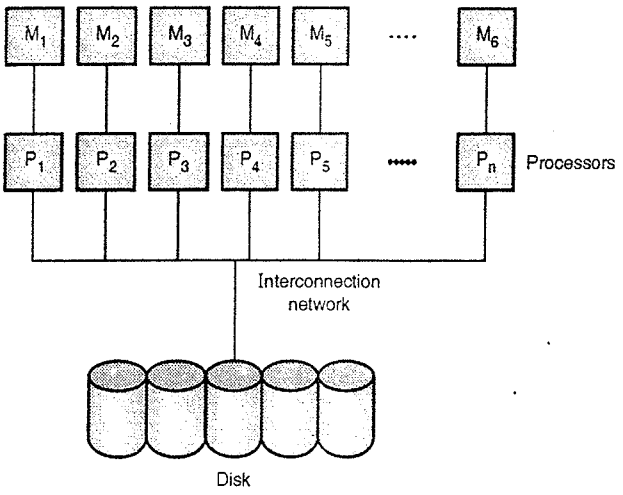


Fig. 5.4.3 : Shared disk architecture

Advantages of shared disk architecture

1. It has fault tolerance means failure of any processor does not lead to execution stop; rather any other processor completes the task.
2. As compared to shared memory architecture it supports large number of processors.
3. This architecture permits high availability.
4. Interconnection to the memory is more smooth and free.

Disadvantages of shared disk architecture

1. The scalability of Shared disc system is limited as large amount of data travels through the interconnection channel.
2. The speed of existing processors may slow down if more processors get added.

5.4.2.3 Shared Nothing Architecture

SPPU - Dec. 15

University Question

Q. Explain Shared Nothing Parallel Database System architecture. **(Dec. 2015, 7 Marks)**

- The shared nothing architecture is a distributed computing architecture in which each node is independent. More specifically, none of the nodes share memory or disk storage.
- In this architecture each processor has its own local memory and local disk.
- Intercommunication channel is used by the processors to communicate.
- The processors can independently act as a server to serve the data of local disk.

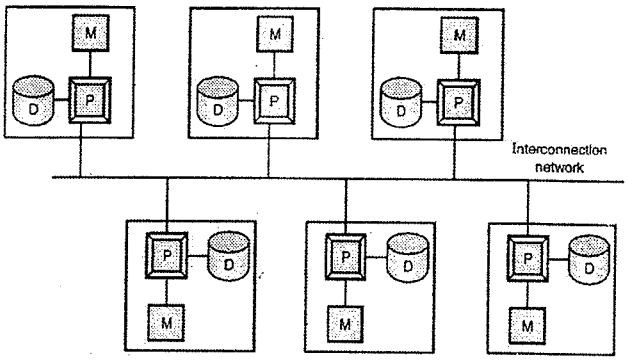


Fig. 5.4.4 : Shared nothing architecture

Advantages of shared nothing architecture

1. This architecture is scalable regarding the number of processors. Increase in their number is easy and flexible.
2. When nodes get added transmission capacity increases.
3. It is a read only database and decision support application.
4. In this architecture failure is local. It means that failure of one node can not affect to other nodes.

Disadvantages of shared nothing architecture

1. The cost of communication is higher than shared memory and shared disk architecture.
2. The data sending involves the software interaction.
3. In this technique more coordination is required.

5.4.2.4 Hierarchical Architecture

Hierarchical model architecture is also known as Non Uniform Memory Architecture. This is nothing but the combination of shared disk, shared memory and shared nothing architecture.

Consider there are two groups of processors say A and B. All processors from both the groups have local

memory. But processors from other groups can access memory which is associated with the other group in coherent.

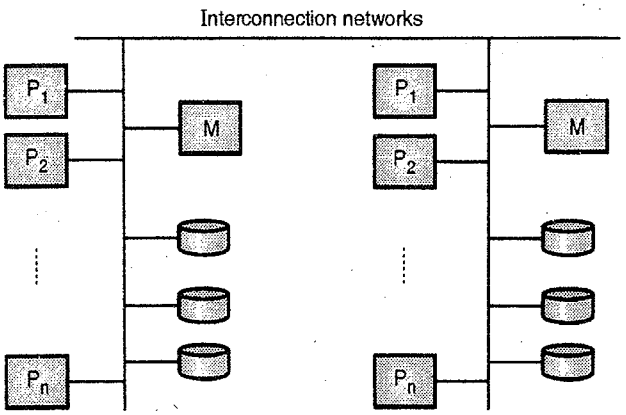


Fig. 5.4.5 : Hierarchical architecture

Advantages of Hierarchical Architecture

1. The availability of memory is more in this architecture.
2. The scalability of system is also more.

Disadvantages of Hierarchical Architecture

1. This architecture is costly as compared to other architectures.

Syllabus Topic : Distributed Database

5.5 Distributed Database

SPPU - Dec. 13, Dec. 15, May 16

University Questions

Q. Write a short note on : Distributed Database System. **(Dec. 2013, 3 Marks)**

Q. Define Distributed Database. **(Dec. 2015, May 2016, 2 Marks)**

5.5.1 Distributed Database Introduction

Distributed database is the type of database management system in which number of databases are stored at various locations and interconnected through the computer networks. Means we can say that, "Distributed database is a collection of various interconnected databases which are physically spread at various locations and communicate via a computer network. In distributed database system each site may have its own memory and its own Database Server.

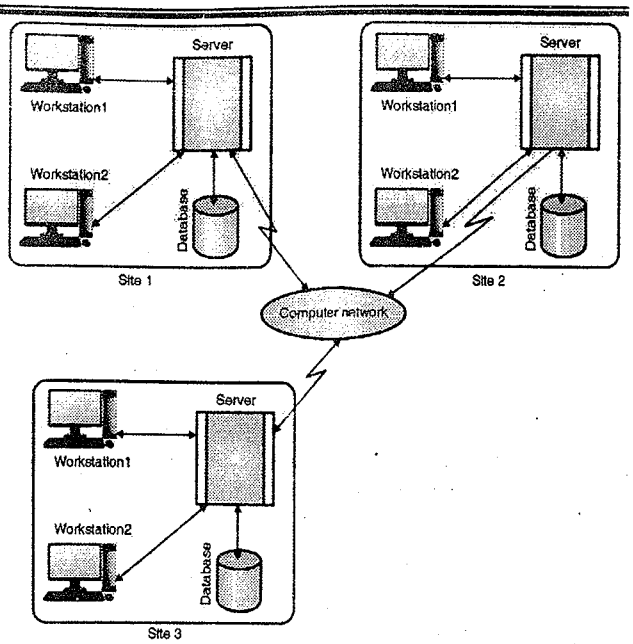


Fig. 5.5.1 : Distributed Database System

5.5.2 Advantages of Distributed Database

SPPU - May 14, Dec. 15

University Question

Q. What are advantages and disadvantages of Distributed Database system architecture? **(May 2014, Dec. 2015, 6 Marks)**

1. **Modular Development :** If we need to expand the same system at new locations, we can use distributed databases, we simply requires to add new computers and local data to the new site and have to connect them to the communication network. It does not interrupt the current functionality.
2. **Increases reliability :** The possibility of the system running at any point is known as reliability. In distributed database system if one component fails to work, other component can take over its function. Due to this the whole system does not affected.
3. **Improve performance :** In distributed database system, large database is distributed among the multiple sites. Due to this distribution a small database is exist at each site. The small database is easy to handle which increases the performance.
4. **Increase availability :** In distributed database system data is scattered at various nodes. Therefore if one node fails then data can be easily available from another node. It means that it have

higher availability.

5. **Faster response :** In this system most of the data is local. Due to this user request can be answered quickly.
6. **Local control :** The sites which contain the data are the owner of the data. These sites can locally control the data.
7. Even though the data is distributed, the system presents the data to the user as if it were located locally. User doesn't know where the data is located physically.

5.5.3 Disadvantages of Distributed Database

1. **Cost :** The requirement of additional hardware to establish a network between sites causes increment of cost. There are ongoing communication costs incurred with the use of this network. There are also additional manpower costs to manage and maintain the local DBMSs and the underlying network.
2. **Security :** In distributed database system database has access from all the sites therefore security becomes an issue for the distributed database system.
3. **Database design more complex :** In distributed database system we have to distribute the large database on various sites in the form of small databases. Due to this, designing of database become more complex.
4. **Increased processing overhead :** It required many messages to share within the sites to complete a distributed transaction.
5. **Complex data integrity :** In distributed database data integrity becomes complex. It requires too many resources to integrate the data.

5.5.4 Types of Distributed Database System

SPPU - Dec. 14, May 16

University Questions

Q. Compare homogeneous and heterogeneous distributed database. **(Dec. 2014, 5 Marks)**

Q. Explain homogeneous and heterogeneous Distributed Databases? **(May 2016, 6 Marks)**

There are two different types of distributed database.

1. Homogenous Distributed Database Systems
2. Heterogeneous Distributed Database Systems

5.5.4.1 Homogeneous Distributed Database Systems

In homogeneous DDBMS, all sites use the same database management system product. In homogeneous distributed database, as all sites have identical software, they are aware of each other and agree to cooperate in processing user requests. A homogeneous DBMS appears to the user as a single system.

It is much easier to design and manage. This design provides incremental growth by making additional new sites to DDBMS easily. It allows increased performance by exploiting the parallel processing capability of multiple sites.

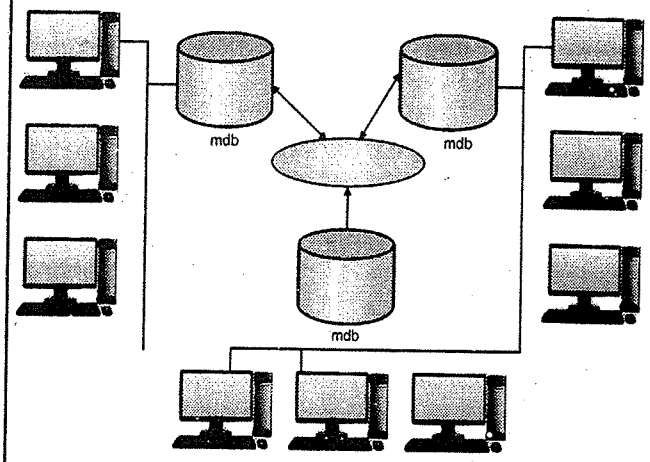


Fig. 5.5.2 : Homogeneous DDBMS

The following conditions must be satisfied for homogeneous database:

- The operating systems used at each location must be same or compatible.
- The database applications used at each location must be same or compatible
- The data structures used at each location must be same or compatible.

5.5.4.2 Heterogeneous Distributed Database System

The word heterogeneous state that the ability to

accept different forms of databases. In heterogeneous database we can use different types of databases. In this type we can use different schemas. In a heterogeneous system, sites may run different DBMS products, which need not based on the same underlying data model, and so the system may be composed of relational, networked, hierarchical and object-oriented DBMSs. In this system, different computers and different operating systems or data models can be used at the each of the location.

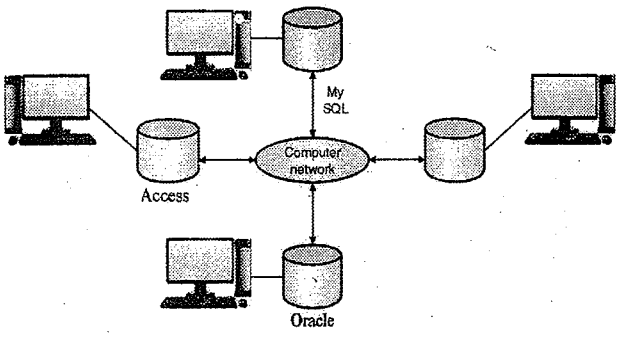


Fig. 5.5.3 : Heterogeneous Database System

Syllabus Topic : Architecture of Distributed Databases

5.5.5 Architecture of Distributed Database

SPPU - May 16, Dec. 16

University Question

Q. Explain distributed database system architecture. **(May 2016, Dec. 2016, 8 Marks)**

DDBMS architectures are generally developed depending on three parameters:

- **Distribution :** It states the physical distribution of data across the different sites.
- **Autonomy :** It indicates the distribution control of the database system and the degree to which each constituent DBMS can operate independently.
- **Heterogeneity :** It refers to the uniformity or dissimilarity of the data models, system components and databases.

Architectural Models

Some of the common architectural models are :

1. Client-Server Architecture for DDBMS
2. Peer-to-Peer Architecture for DDBMS
3. Multi-DBMS Architecture

5.5.5.1 Client-Server Architecture for DDBMS

Client-server architecture is a two-level architecture where the functionality of architecture is divided into servers and clients. Data management, query processing, optimization and transaction management are the functions of server. Client functions include mainly user interface, consistency checking and transaction management.

The two different client-server architectures are :

1. Single Server Multiple Client
2. Multiple Server Multiple Client

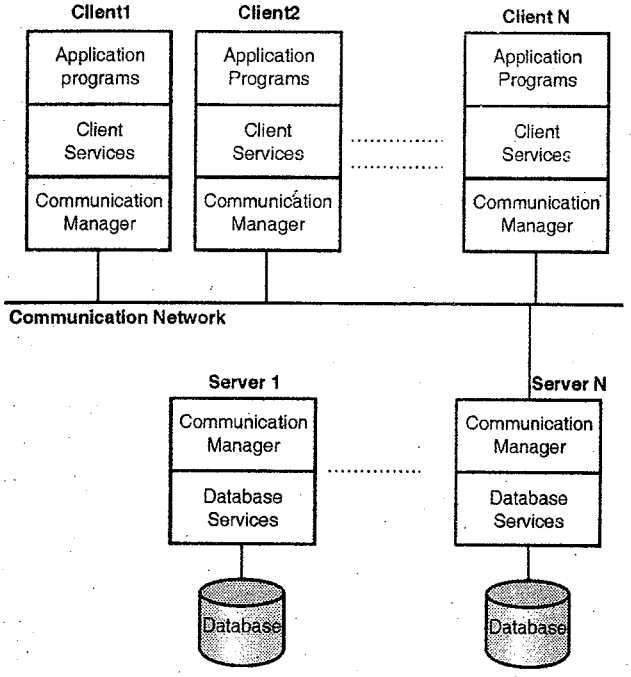


Fig. 5.5.4 : Client-server architecture

5.5.5.2 Peer-to-Peer Architecture for DDBMS

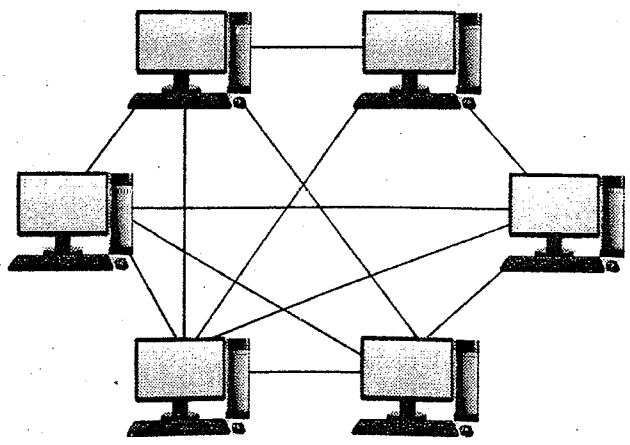


Fig. 5.5.5 : Peer-to-peer architecture for DDBMS

- The network architecture in which each workstation or node has same capabilities and

responsibilities is known as peer-to-peer architecture. Peer-to-peer may also be used to refer single software design. Due to this each peer may act as both client and server.

- The peers share their resources with other peers and co-ordinate their activities.

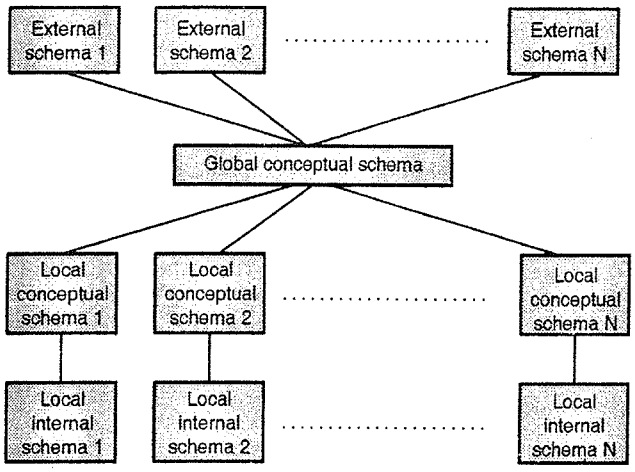


Fig. 5.5.6

- This architecture generally has four levels of schemas
  1. **Global Conceptual Schema :** As the name suggest, it shows only the global logical view of data.
  2. **Local Conceptual Schema :** It only shows logical data organization at each site. The data on every site is local for that particular site.
  3. **Local Internal Schema :** Local internal schema shows the physical data organization at each and every site.
  4. **External Schema :** External schema shows the external user view.

5.5.5.3 Multi - DBMS Architectures

Multi-database management system architecture is an integrated database system formed by collection of two or more autonomous database systems. Multi-DBMS can be expressed through six levels of schemas.

- **Multi-database View Level :** It shows multiple user views which consists of subsets of the integrated distributed database.
- **Multi-database Conceptual Level :** It shows the integrated multi-database that consists of global logical multi-database structure definitions.
- **Multi-database Internal Level :** It shows the data distribution across different sites and multi-database to local data mapping.
- **Local database View Level :** It shows the public view of local data.
- **Local database Conceptual Level :** It shows the local data organization at each site.
- **Local database Internal Level :** It shows the physical data organization at each site.

There are two design alternatives for multi-DBMS

- Model with multi-database conceptual level.

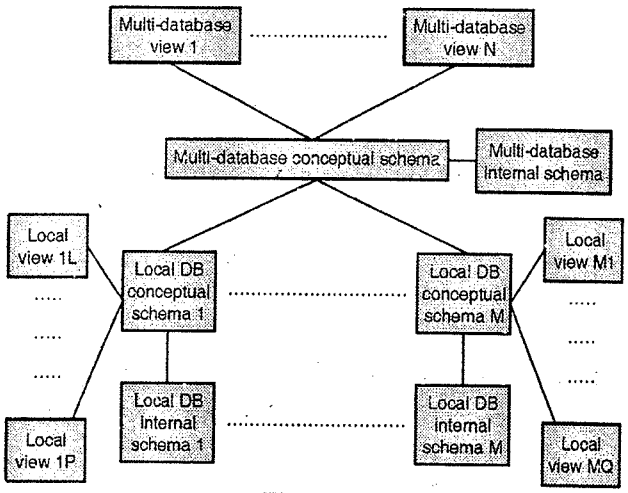


Fig. 5.5.7

- Model without multi-database conceptual level.

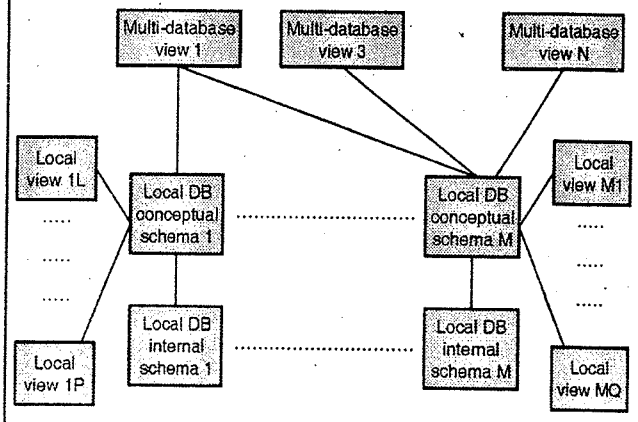


Fig. 5.5.8

Syllabus Topic : Distributed Database Design

5.5.6 Distributed Database Design

The design of distributed database includes

A) Design problem
B) Design strategies(top-down, bottom-up)
C) Distributed Database Storage
- Fragmentation
- Data replication
- Data Allocation

5.5.6.1 Design Problem

In **designing a distributed database**, you must decide which portion of the database and programs are to be stored at which location. It also includes the designing of network itself.

- In designing of distributed database the distribution of application involves
  1. DDBMS software distribution
  2. Distribution of application that run on the database
- Important points for the analysis of distributed systems are -
  - The level of sharing - data sharing, data + program sharing or no sharing
  - Behaviour of access patterns - static or dynamic
  - Level of knowledge on access pattern behavior - partial information, complete information or no information.

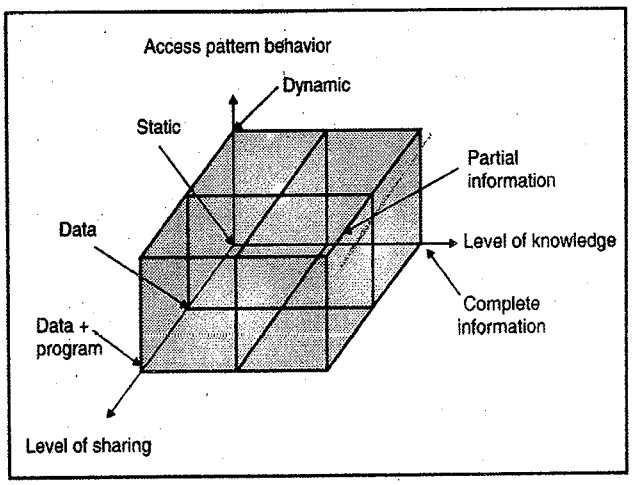


Fig. 5.5.9

5.5.6.2 Design Strategies

Top-down approach

- Designing systems from initial stage
- Homogeneous systems

Bottom-up approach

- The databases considered to be exist at a number of sites already
- To solve common tasks the databases should be connected

Syllabus Topic : Distributed Data Storage

5.5.6.3 Distributed Data Storage

To store a relation in distributed database design there are following approaches

A) Fragmentation
B) Data Replication
C) Data Allocation

A) Data Fragmentation

The data fragmentation is the technique used in distributed database design. This technique is used to break up the database into logical units called **fragments**. The information of fragmentation is stored into the catalogue of distributed database system which is used by the processing computer to process the user's request. There are three different types of fragmentation :

1. **Horizontal Fragmentation :** The fragmentation of relation is done horizontally, in the form of rows. Each fragment which contains unique rows is stored in different computer nodes. Each horizontal fragment may have a different number of rows, but each fragment must have the same number of attributes.
2. **Vertical Fragmentation :** The division of relation into fragments consist of collection of attributes. In the vertical fragments there can be same number of rows and can have different attributes which depends upon the key.
3. **Mixed Fragmentation :** Mixed Fragmentation is the combination of both horizontal and vertical fragmentation. It is the two-step process. The first step is to achieve the horizontal fragmentation to obtain the necessary rows. And second step is to achieve vertical fragmentation to divide attributes among the rows.

B) Data Replication

The data replication is the storage of copies of data at multiple sites on the network. Fragmented copies can be stored at various sites. The data replication enhances data availability and response time. All copies of fragmented data must be identical. The maintenance of replication may become complex. A database can be either **fully replicated**, **partially replicated** or **un-replicated**. We can use data replication while we have to handle large database. Due to replication it becomes possible to retrieve the lost data. We can retrieve lost data from the other sites.

1. **Fully replicated :** The fully replicated database stores multiple copies of entire database on all the sites. The availability of entire database on all the sites leads to fast processing of queries.
2. **Partially replicated :** Portion of tables(relations) is stores on different sites. The frequency of data access is considered while distributing the data on different sites.
3. **Non replicated or No replication :** Non replicated replication stores single copy of database fragment at a single site. There is no duplication occurs in this phase.

C) Data Allocation

The data allocation decides the locations of different data for storage purpose. Data allocation can be centralized, partitioned or replicated.

1. **Centralized :** The data can be stored on a single particular site. There is no distribution of database.
2. **Partitioned :** The database get divided into multiple fragments and stored on various sites.
3. **Replicated :** Copies of one or more database fragments are stored at several sites.

Syllabus Topic : Distributed Transaction Basics

5.6 Distributed Transaction

A **distributed transaction** contains one or more statements which make updations in the data of two or more distinct nodes of a distributed database.

5.6.1 Distributed Transaction Basics

There are two types of transaction we can consider :

1. **Local Transaction :** Local transactions are the transactions which perform operation only on the single local database. These transactions can access and update the data from local database only.
2. **Global Transaction :** Global transactions are the transactions which perform operation on the several local databases. It can access and update several local databases.

For distributed transaction, we will study the system structure of distributed database and its possible failure modes. Maintaining ACID properties of transaction is very important task which is done by a utility called as Transaction Manager. All sites have their own Transaction Managers which co-operate each other to execute the transactions successfully.

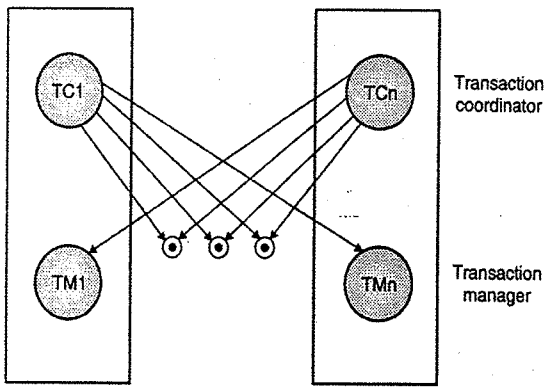


Fig. 5.6.1

Now to understand the working of Transaction Manager we will consider a system with two subsystems.

- **Transaction Manager :** The execution of transactions which access the local data is managed by the Transaction Manager. These transactions may be local or part of any global transaction. The Transaction Manager should participate in appropriate concurrency control scheme to coordinate the concurrent execution of transactions executing that site.
- **Transaction Coordinator :** At the site, the execution of all local and global transactions is coordinated by the Transaction Coordinator. The Transaction coordinator should start the execution of transaction.

Break the transaction in sub-transactions and assign it to different sites. Coordinate the completion of transaction, in which the transaction either committed or aborted.

Syllabus Topic : Failure Modes

5.6.2 Failure Modes

In distributed transactions failure can be broadly categorized into three different groups. They are as follows:

1. Soft Failure

This type of failure can cause the loss of volatile memory of the computer. Due to this we lost the information stored in the non-persistent storage like main memory, buffers, caches or registers. It is also known as system crash. The various types of soft failures are as follows :

1. Crash of Main memory
2. Transaction failure or abortion.
3. Power failure.
4. Integer overflow or divide-by-zero exceptions.
5. Failure of supporting software.
6. Operating system failure.

2. Hard failure

Hard failure causes the loss of data which is stored in non-volatile storage such as Disk. It is also known as Disk Failure. Because of failure in disk, corruption of data in some disk blocks or failure of the total disk may occur. The causes of a hard failure are as follows :

1. Faults in media.
2. Malfunction in Read-write operations
3. Information Corruption on the disk.
4. Power failure.
5. Read/write head crash of disk.

If we have new, formatted and ready-to-use disk in backup then recovery from disk failure can be easy and short.

3. Network Failure

Network failures are widely spread in distributed or network databases. It consist of the errors induced in the database system due to the distributed format of the data in the network and transformation of data.

There are number of reason of network failure

1. Failure in communication link
2. Data corruption in transformation process.
3. Site failures.
4. Congestion in network

Syllabus Topic : Commit Protocols

5.7 Commit Protocols

Atomicity is an important ACID property. To maintain atomicity it is necessary that the final outcome of any transaction should be accepted by all the sites on which it is executing. That means the transaction must be either committed for all the sites or aborted for all the sites. For this purpose the commit protocols are used.

There are different commit protocols. They are as follows:

1. One-phase Commit Protocol (1 PC)
2. Two-phase Commit Protocol (2 PC)
3. Three-phase commit protocol (3 PC)

5.7.1 One-phase Commit Protocol (1 PC)

One phase commit protocol in distributed database is the simplest commit protocol. In distributed transaction there is a controlling site and a number of slave sites where the transaction is being executed. For the One phase commit protocol following steps takes place:

1. When each slave has completed its transaction locally, it sends a done message to the controlling sites.
2. After sending the message the slave sites wait for 'Commit' or 'Abort' message from controlling sites. This waiting time is called as window of vulnerability.

3. When a controlling site receive 'Done' message from slave site, it takes a decision to commit or abort. This is called as Commit Point. Then the controlling site sends this decision to all the slave sites.
4. After receiving this message from controlling sites, the slave sites either commit or abort and then send an acknowledgement message to controlling sites.

5.7.2 Two-phase Commit Protocol (2 PC)

SPPU - Dec. 15

University Question

Q. Explain Two Phase Commit Protocol in Distributed Database **(Dec. 2015, 5 Marks)**

In distributed database **two phase commit protocols** reduces the waiting time means vulnerability of one phase commit protocol. Two steps are performed in two phase commit protocol as follows:

Step 1: Prepare Phase

- As we have seen in one phase protocol, after completing the transaction locally, each slave sends a 'done' message to the controlling sites. When the controlling sites received 'done' message from all slaves. It sends a prepare message to the slave sites.
- The slaves sites vote on whether they want to commit or not. If a slave wants to commit, it sends a "Ready" message.
- A slave site that does not want to commit, sends a "Not Ready" message to controlling sites. This may happen when the slave site has conflicting concurrent transactions or there is a timeout.

Step 2 : Commit/Abort Phase

- After the controlling site received "Ready" message from all the slave sites :
  - The controlling site sends a "Global Commit" message to the slave sites.
  - The slave sites apply the transaction and send a "Commit ACKNOLEDEGEMET" message to the controlling site.
  - When the controlling site receives "Commit Acknowledgement" message from all the slaves, it considers the transaction as committed.
- After the controlling site has received the first "Not Ready" message from any slave :
  - The controlling site sends a "Global Abort" message to the slaves.
  - The slaves abort the transaction and send "Abort Acknowledgment" message to the controlling site.
  - When the controlling site receives "Abort Acknowledgment" message from all the slaves, it considers the transaction as aborted.

5.7.3 Three-phase Commit Protocol (3 PC)

SPPU - Dec. 15

University Question

Q. How 3 PC is different than 2 PC? **(Dec. 2015, 3 Marks)**

This protocol contains one more phase '**Prepare to Commit Phase**' than the **2 PC**. We will see the **difference** between these two protocols in the three steps of **3 PC**.

Step 1 : Prepare Phase

- As we have seen in one phase protocol, after completing the transaction locally, each slave sends a 'done' message to the controlling sites. When the controlling sites receives 'done' message from all slaves. It sends a prepare message to the slave sites.
- The slaves sites vote on whether they still want to commit or not. If a slave wants to commit, it sends a "Ready" message.
- A slave site that does not want to commit, it sends a "Not Ready" message to controlling sites. This may happen when the slave site has conflicting concurrent transactions or there is a timeout.

Step 2 : Prepare to Commit Phase

- In this step controlling site issues an "Enter Prepared State" broadcast message.
- The slave sites vote "OK" in response.

Step 3 : Commit / Abort Phase

- After the controlling site receives "Ready" message from all the slave sites–
  - The controlling site sends a "Global Commit" message to the slave sites.

- The slave sites apply the transaction.

– After the controlling site has received the first "Not Ready" message from any slave –

- The controlling site sends a "Global Abort" message to the slaves.
- In this phase commit acknowledgement an abort acknowledgement is not required unlike two phase protocol.

Syllabus Topic : Concurrency Control in Distributed Database

5.8 Concurrency Control in Distributed Database

In distributed database systems, database is typically used by many users. These systems usually allow multiple transactions to run concurrently i.e. at the same time. The activity of coordinating concurrent access to a single database in a multiuser database management system (DBMS) is known as **Concurrency control**. It allows users to access a database in a multi-programmed fashion. In centralized database system, several locking protocols are used for concurrency control. The major difference between centralized and distributed database system is that how lock manager deal with replicated data.

There are some approaches for concurrency control in distributed database :

A) Single Lock Manager

- In distributed database system which have several sites, maintains a single lock manager at a particular chosen site this concept is known as single lock manager approach. Fig. 5.8.1 shows the single lock manager approach.
- Here the sites S1, S2, S3, S4, S5, and S6 are present among those the data D is replicated on S1 S6 and S4. And the site S3 acts as a Lock Manager.
- At the site S2 transaction T1 requests for data item D(step 1). These requests are forwarded by the site S2 to the lock manager's site S3 for granting purpose (step 2).

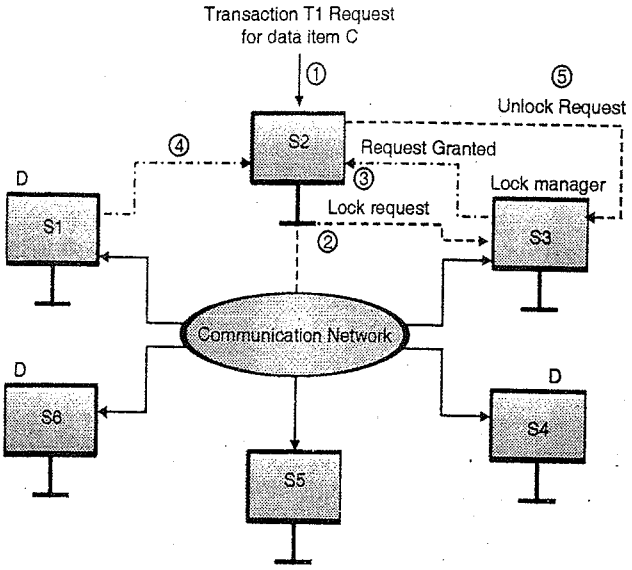


Fig. 5.8.1 : Single lock manager approach

- If the requested data item is free then the request for that data item is granted immediately by the lock manager(step 3).
- If the request is granted then the data item can be read from any site where the replica of required data item is present; here, the site S2 takes data item D from site S1(step 4).
- After the transaction T1 executed successfully the Transaction /manager at site S2 again send the request for unlock the data item so that other transactions can used the data item D now(step 5).

B) Distributed Lock Manager

- In distributed database system there are several sites and data is replicated or fragmented on those sites. The distributed lock manager approach is nothing but the distributing the functionality of lock manager over several sites.
- Fig. 5.8.2 shows the distributed lock manager approach using primary copy protocol.
- Here the sites S1, S2, S3, S4, S5, and S6 are present.
- Data A, B, C is replicated on multiple sites.
- Among the six sites S6 holds primary copy of data item B so the lock manager for granting data access on B is present at only site S6, similarly S5 site acts as a lock manger for data item A and S4 acts as a lock manager for data item C.

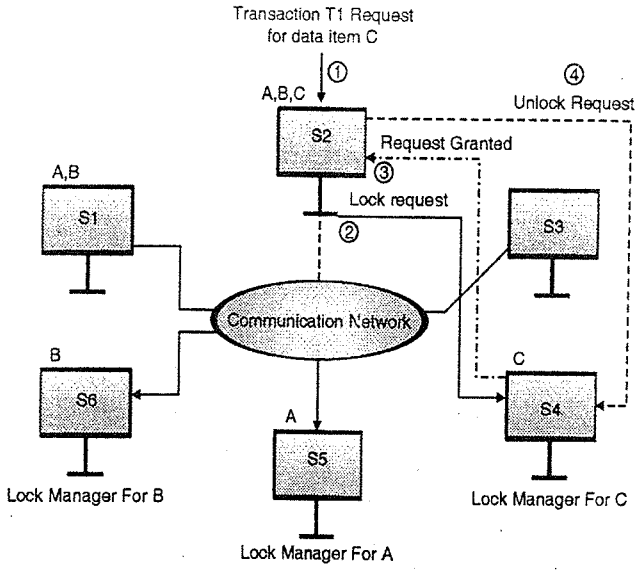


Fig. 5.8.2 : Distributed lock manager approach

- At the site S2 transaction T1 requests for data item C (step 1). Even if the replica of C is present at site S2 locally it is mandatory to first the request lock for accessing C from site S4 as it is acts as lock manager for C. So the transaction manager at site S2 sends lock request to site S4(step 2).
- If the requested data item is free then the request for that data item is granted immediately by the lock manager, hence site S4 grants lock request for C(step 3).
- Now the transaction T1 can access local copy of Data item C present at site S2(step 4).
- After the transaction T1 executed successfully the Transaction manager at site S2 again send the request for unlock the data item so that other transactions can used the data item C now(step 5).

❑❑❑

UNIT VI

CHAPTER 6

NoSQL Database

Syllabus

- Introduction to NoSQL Database
- Types and examples of NoSQL Database- Key value store, document store, graph, Performance
- Structured verses unstructured data
- Distributed Database Model
- CAP theorem and BASE Properties
- Comparative study of SQL and NoSQL
- NoSQL Data Models
- Case Study - unstructured data from social media
- Introduction to Big Data, Hadoop : HDFS, MapReduce

Syllabus Topic : Introduction to NoSQL Database

6.1 Introduction to NoSQL Database

The relational databases are widely used in software industry. The design of relational databases is not such that which can cope with the scale and agility challenges that face modern real time applications, nor were they built to obtain benefit of the commodity storage and processing power available now a day. Now a days the data management becomes very difficult because of the tremendously increase in the size of data in various emerging fields like social networking, e-commerce etc.

- NoSQL is also known as "non SQL" or "non relational" or "Not Only SQL". NoSQL is database which provides a complete different mechanism for storing and retrieval of data which is modelled in means other than the tabular relations used in relational database management systems.
- Sometimes the term NOSQL seems to be confusing as handling data without SQL is out of imagination for some people. But actually the meaning of SQL is Not Only SQL.
- NoSQL challenges the dominance of relational databases. NoSQL databases are increasingly used in big data and real-time web applications.
- In NoSQL the data structures used like key-value, column, graph or document are different from those used in traditional relational database system which makes some operations faster in NoSQL.
- Usually the data structures used by NoSQL databases are also more flexible than relational database system.
- NoSQL allows the insertion of data without a predefined schema (design). In this database, it is possible to make significant application changes in the system without having worry about service interruptions. Because of it, the speed of development increases, the integration of code becomes more reliable and time of database administration decreases.
- To enforce data quality controls, developer has to add application-side code. NoSQL supports validation rules to be applied on the database which helps user to control the data while maintaining the advantage of a dynamic schema.
- NoSQL databases are more concentrated on availability, partition tolerance, and speed for which they may compromise the consistency. The

basic reason of no wide adoption of NoSQL is use of low level query language rather than SQL.

History of NoSQL

- Such database have existed since year 1960, but not known as "NoSQL". Need of this database comes in picture with the rise of web related applications like Google, Facebook, Amazon etc.
- In 1998 Carlo Strozzi first introduced a lightweight, open source relational database system which did not expose the standard Structured Query Language (SQL) interface. Carlo Strozzi gives the name as "NoSQL" to it. He suggested that as the NoSQL is differ from the traditional relational model, it should be called as "NoREAL" means "No Relational".
- Ohan Oskarsson, then a developer at Last.fm, reintroduced the term *NoSQL* in early 2009 when he organized an event to discuss "open source distributed, non relational databases". The name attempted to label the emergence of an increasing number of non-relational, distributed data stores, including open source clones of Google's BigTable/ MapReduce and Amazon's Dynamo.
- Most of the early NoSQL systems did not attempt to provide atomicity, consistency, isolation and durability guarantees, contrary to the prevailing practice among relational database systems.
- Based on 2014 revenue, the NoSQL market leaders are MarkLogic, MongoDB, and Datastax. Based on 2015 popularity rankings, the most popular NoSQL databases are MongoDB, Apache Cassandra, and Redis.

Use NoSQL when needs are like

1. Decentralized applications (e.g. Web and mobile)
2. Continuous availability; no downtime
3. High velocity data (devices, sensors, etc.)
4. Data coming in from many locations
5. Structured data is available with some semi/unstructured data.
6. To maintain high data volumes; retain forever.

What is NoSQL ?

- NoSQL systems are also called as "Not only SQL" which indicates that they may support query languages like SQL.
- NoSQL is not depending on column, rows or schema for structure, it is no-relational database management systems. The data models of NoSQL are more flexible.
- NoSQL provides scalability, availability and fault tolerance and emerges as a alternative for relational database.
- The speciality of NoSQL is that, it may not require fixed table schemas avoids join operations, and also scale horizontally.
- Developers are working with applications that create massive volumes of new, rapidly changing data types - structured, semi-structured, unstructured and polymorphic data. NoSQL is useful for data which is growing far more rapidly or unstructured data or data which does not store in the relational schemas of RDBMS. There are common types of unstructured data: user and session data; chat, messaging, and log data; time series data such as IoT and device data; and large objects such as video and images.

Features of NoSQL

- Design simplicity.
- Simpler "horizontal" scaling to clusters of machines. This was a problem in relational databases.
- More control over data availability.

Observations regarding NoSQL

- It does not use the relational model.
- It runs well on clusters.
- NoSQL is Mostly open-source.
- It is Schema-less.

Why NoSQL ?

- All IT professionals and industry database experts come to know that NoSQL is here to stay.
- A recent study performed on NoSQL market growth forecasts a very strong compound annual growth rate of 21 percent for NoSQL technology from 2013-2018. It shows bright future for NoSQL.

- NoSQL databases has been much more clearly articulated today. NoSQL database refers to groups of databases that are not based on relational database model.
- The data storage model used by NoSQL database is not some fixed data model, but the common features among the NoSQL database is that the relational and tabular database model of SQL based database is not used.

Advantages of NoSQL

We cannot consider that NoSQL databases are straight substitution for relational database management system (RDBMS). But for many issues regarding data, NoSQL seem to be better.

1. **Data storage :** NoSQL databases supports storing data "in the form of Key value pair which give ability to store simple data structures. The document NoSQL database provides the ability to handle a range of flat or nested structures.
2. **Support for unstructured text :** NoSQL databases can handle unstructured text easily. This ability increases information effectively and can help organizations make better decisions.
3. **Ability to handle change over time :** NoSQL databases are capable of managing changes because of the systematic storage system.
4. **No reliance on SQL magic :** SQL(Structured Query Language) is the predominant language which is used to write queries in relational database management systems. Even if several NoSQL databases provide support for SQL access, they do so for compatibility with existing applications like business intelligence (BI) tools. NoSQL is not dependent on SQL for processing. NoSQL databases support their own query languages that can support data processing.
5. **Ability to scale horizontally on commodity hardware :** NoSQL databases supports distribution of a database across several servers. Hence if there is requirement of more data storage, then number of servers can be increased and connect them to database cluster (horizontal scaling) making them work as a single data service.
6. **Breadth of functionality :** Near about all the relational databases support the same characteristics but in a slightly different way, so they are all similar. In contrast, the NoSQL databases come in different core types: key-value, document store and graph. Out of these types, the one select to suit our requirements is not hard.
7. **Support for multiple data structures :** There is requirement of simple as well as complex data structures. NoSQL databases provide support for a range of data structures. Key-value stores can handle Simple binary values, lists, maps, and strings. Document databases can manage highly complex parent-child hierarchal structures Graph stores can describe the web of interrelated information.
8. **Big data applications :** In some systems the data grows rapidly. Such big volume data can be easily handled by NoSQL databases.
9. **Database administration :** The NoSQL has data distribution and auto repair capabilities, simplified data models and fewer tuning and administration requirements. This leads to less requirement of hands-on management.
10. **Economy :** These databases are designed to be used with low-cost commodity hardware.

It is difficult for application developer to find match between the relational data structures and the in-memory data structures. Using NoSQL databases they can develop the system without having to convert in-memory structures to relational structures.

Disadvantages of NoSQL

1. No standard schema.
2. Less use of SQL.

Companies using NoSQL

Now a day's many companies using NoSQL. Some of them are :

- Google
- Facebook
- Adobe
- Foursquare
- Digg
- Vermont public radio
- LinkedIn
- Mozilla

Syllabus Topic : Types and Examples of NoSQL Database

6.2 Types and Examples of NoSQL Database

There have been various approaches to classify NoSQL databases, each with different categories and subcategories.

Following are some types of NoSQL :

1. Key Value Store
2. Document Store
3. Column Store
4. Graph

Syllabus Topic : Key Value Store

6.2.1 Key Value Store SPPU - Oct. 16

University Question

Q. Explain key value store NOSQL data model. **(Oct. 2016(In sem), 5 Marks)**

- The Key-value (KV) store system uses the concept of associative array, as their fundamental data model. In this model, data is represented as a collection of key-value pairs. Every single item in the database is stored as an attribute name (or 'key'), together with its value.
- In NoSQL database, a table exists with two columns : one is the Key and the other is Value.
- The key in the key-value pair should not be repeated because it the unique identifier that helps to access the value associated with that key uniquely.
- The Key value stores allow the developer of application to store schema-less data.
- Key-value databases are the simplest form of the NoSQL databases. Other advanced models are mostly extensions to this key-value model.
- **Examples** include Memcached, Riak, ArangoDB, InfinityDB, Oracle NoSQL Database, Redis and dbm.
- All key-value databases are not similar; there are major differences between these databases. For example: Data in **MemcacheDB** is not persistent while in **Riak** it is persistant. Such features are useful when implementing some solutions. It is important to not only choose a key-value database based on your requirements, it is also important to choose which key-value database.
- Examples of key-value NoSQL database applications :
  - Dynamo
  - MemcacheDB
  - Redis
  - Riak
  - FairCom
- Four main core operations perform on key-value store :
  1. **Get(key) :** It return the single value of given key.
  2. **Put(key,value) :** It assigns value to key.
  3. **Multi-get(key1,key2,key3,..,keyn):** It returns multiple values of given multiple keys.
  4. **Delete(key) :** It deletes both key and value present in it.

Example

This is a simple example for key-value store. Here keys are the names of employees and values are their contact numbers.

Key	Value
Sam	(234) 567-8901
jack	(134)526-6845
Ron	(245)452-4584
kenny	(356)584-1458

Where key value store is used ?

Key-value databases can be used in many scenarios. Such as,

- **General Web / Computers**
  - User profiles
  - Article/blog comments
  - Emails
- **E-commerce**
  - Shopping cart contents
  - Product categories
  - Product details

Advantages of Key Value Store

- Key value store is the simplest type of NoSQL.
- Supports simple queries very efficiently.

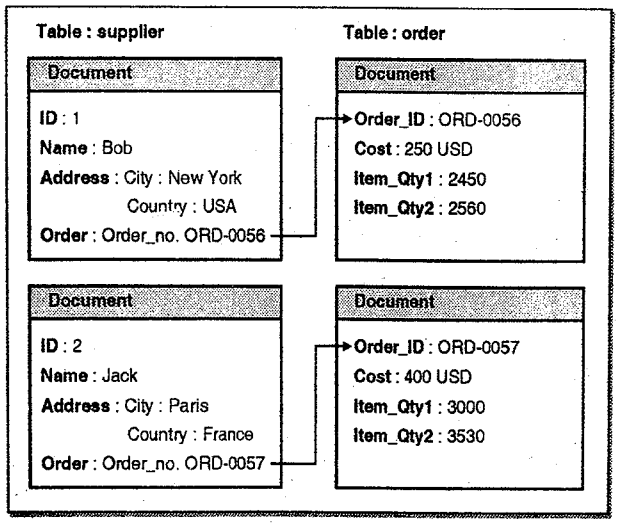
- Extended form of key-value stores is able to sort the keys.
- It is specially designed for storing data as a schema free data.
- Very simple data-modeling pattern should be understandable by anyone.
- With little or no maintained indexes, the key-value stores are designed to be more scalable and extremely fast.
- Suitable for system where data is not highly related.

Disadvantages of Key Value Store

- The indexing and scanning capabilities are absent. It does not help if we want to perform more operation as per user requirement than the basic CRUD (Create, Read, Update, Delete) operations.
- Only one row simple queries can be executed efficiently.
- Difficult to perform SQL operations like JOINS, GROUP BY etc.
- Selecting appropriate data type for value is difficult.
- Difficult to use constraints like FOREIGN KEY or NOT NULL.
- More application code is required to reassemble collections of key-value pairs into objects.

Syllabus Topic : Document Store

6.2.2 Document Store



- These databases store records as "documents" where a document can generally be thought of as a grouping of key-value pairs.
- The documents are identified by the unique keys which represents them. One defining characteristics of a document-oriented database is that in addition to the key lookup performed by a key-value store, the database offers an API or query language that retrieves documents based on their contents.
- Document databases are extension to key-value store. Addition to query capabilities of key-value databases, they provide indexing and the ability to filter documents based on attributes in the document.
- Examples of Document Store NoSQL database applications :
  - o MongoDB
  - o Couchbase

Advantages of Document Store

- The performance is good and the distribution across various servers becomes a lot easier.
- There is no need of translation between object in SQL and application. The object can directly be converted into document.
- They have strong indexing features and can rapidly execute different queries.

6.2.3 Column Store

- Column store is column oriented NoSQL database. Data is stored in cells grouped in columns of data rather than as rows of data. The logical grouping of columns is created in column families. There is no limitation on number of columns in a column family.
- These columns can be created at runtime. The operations like read write are performed using columns rather than rows.
- The column store structure gives the advantage of fast search / access and data aggregation over the row format data storage of relational databases.
- Relational databases store a single row as a continuous disk entry. Different rows are stored in different places on disk while column store database store all the cells related to a column as a

continuous disk entry which makes the search/access faster.

- **For example :** Displaying titles from a bunch of a million articles will be a tedious and time wasting task while using relational databases as it will go over each location to get item titles. While in column store, title of all the items can be obtained with just one disk access.
- Examples of column store NoSQL database applications :
  - o HBase
  - o BgTable
  - o HyperTable

Advantages of Column Store

- Efficient storage and data compression.
- Fast data loads.
- Simple configuration.

Disadvantages of Column Store

- Queries with table joins can reduce high performance.
- Transactions are to be avoided or just not supported.

Syllabus Topic : Graph

6.2.4 Graph Store

- The Graph Store NoSQL database technology is designed to handle very large sets of data which may be structured, semi-structured or unstructured.
- In a Graph Base NoSQL Database, rigid format of SQL or the tables and columns representation does not exists. Rather a flexible graphical representation is used which is perfect to address scalability concerns. The graph structure contains edges, nodes and properties.
- The graph store is helpful in transferring the data from one model to other very easily.
- Graph store stores the entities and relationships between these entities. Entities are also known as nodes, which have properties. Nodes represent entities such as people, businesses, accounts, or any other item to be tracked.
- They are roughly the equivalent of the record, relation, or row in a relational database, or the document in a document database.
- Relations are known as edges that can have properties.

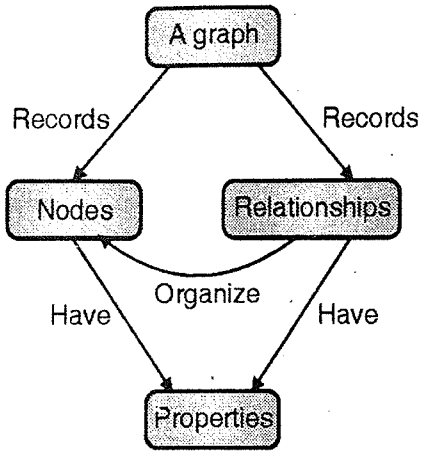


Fig. 6.2.1

- Edges are denoting directions. Nodes are generally organized with the help of relationships. The data is stored once by the organization of graph. This data can be interpreted in different ways depending upon the relationships between nodes.
- Key points related to graph store.
- Edges and nodes are used by these databases to represent and store data.
- The nodes are organised with the help of some relationships in between them, which is represented by edges between the nodes.
- Both the nodes and the relationships have some definite properties.
- Examples of Graph store NoSQL database applications :
  - o Neo4j
  - o Polyglot

Advantage of Graph Store

- Performance
- Flexibility
- Agility(ability to move quickly or easily)

Comparison of all the four NoSQL Databases

Database model	Performance	Scalability	Flexibility
Key value store database	High	High	High
Column store database	High	High	Moderate
Document store database	High	Variable(high)	High
Graph database	Variable	Variable	High

Syllabus Topic : Performance

6.3 Performance

- NoSQL is basically an advanced database developed to deal with the limitations of traditional relational databases in meeting Big Data demands. NoSQL represent various technologies which consider the three V's of big data.
  1. The exponentially growing volume of data.
  2. The velocity in which this data needs to be processed.
  3. The great variety of data being generated by today's applications.
- These demands require a high performance database.
- The data models of NoSQL and RDBMS are different. As we have seen NoSQL systems can be divided into four distinct groups :
  1. Key-value stores
  2. Document-oriented stores
  3. Column family stores
  4. Graph Store
- The database of NoSQL is distributed on different machines. The data models of key-value stores, document stores, and column family stores are key oriented. Hence there are two partition (distribution) strategies which are based on keys.
  1. In the first strategy, the datasets are distributed by the range of their keys. The keyset is split into blocks by the routing server. These blocks are allocated to different nodes. Then one node is assigned to handle the specific key range. Client contacts the routing server to find certain key to get the partition table.
  2. The second strategy is used to provide more availability and simpler cluster architecture. The performance of NoSQL increases because of load balancing. Because of replication, the failing nodes can be easily replaced which provider better availability and durability.
- Performance is an important factor. To evaluate performance of data storage solutions, specific scenarios are used. The general operations performed by applications that use the data store are simulated by these scenarios.
- There are certain advantages of NoSQL which effectively enhance its performance.
- Expressive query language which allows developers to build powerful features with data.
- Secondary indexes offer powerful indexes for quick data navigation along with the benefits of NoSQL.
- Scalability across inexpensive commodity hardware so one can easily grow the application to meet rising demands.
- Flexible data model that allows to quickly adapt the changing needs of your business.

Syllabus Topic : Structured verses Unstructured Data

6.4 Structured verses Unstructured Data

Now a day, the Structured and unstructured data are both used effectively in big data analysis. Few years ago processing capability was limited, memory was inadequate, and cost of data-storage was high. Hence, structured data was the only option to manage data effectively. In recent times, the use of unstructured data increases due to increased availability of storage and the sheer number of complex data sources.

6.4.1 Structured Data

- It is considered that about twenty percent of data in business transactions is in structural form. This structured data is easy to store and manage in

different databases. The structured data can be easily ordered and processed by data mining tools.

- Structured data is the traditional data which consist of very well-organized information. It concerns all data which can be stored in database in the format of tables with rows and columns.
- Sometimes the structured data can be machine generated. For example data regarding heart rate, blood pressure that comes from medical devices, data like rotation per minute, temperature which come from manufacturing sensors or web server logs (number of times a page is visited).
- The Structured data can also be generated by human. For example personal information of an employee line id, name salary etc.
- The data has the advantage of being easily entered, stored, queried and analyzed. The structured data is always easy to compare For Example - Salary of Manager is more than the Clerk. Kunal got more marks than Rahul.
- Some more examples of structured data :
  - Customer data
  - Sales data

6.4.2 Unstructured Data

- The unstructured data is the form of information which does not have any predefined data model. Simply the unstructured data means the data which is not well defined. This data is not suitable for traditional relational model. It is usually text-heavy, but also contains data like as dates, numbers etc. Such data is difficult to understand and manage using traditional programs because it generates irregularities and ambiguities. In social media maximum of data is in unstructured form. The most big data sources like WhatApp, Facebook, twitter have unstructured data. It is difficult to run any analytics on such data. To analyze such data we need to convert it into structured form.
- In 1998, Merrill Lynch cited a rule of thumb that somewhere around 80-90% of all potentially usable business information may originate in unstructured form. This rule of thumb is not based on primary or any quantitative research, but nonetheless is accepted.
- Data with some form of structure may still be characterized as unstructured if its structure is not helpful for the processing task easily.
- For unstructured data RDBMS is not suitable or it is not fit in it.
- The example of textual unstructured data is email messages, PowerPoint presentations, Word documents, chat data etc. Non-textual unstructured data is in the form like JPEG images, MP3 audio files and Flash video files.

6.4.3 Comparison between Structured and Unstructured Data

Sr. No.	Structure Data	Unstructured Data
1	Structured data contain well defined content.	Unstructured data not containing well defined data.
2	Structured data is easily understood.	Unstructured data has to be processed to understand.
3	Data is stored in RDBMS.	RDBMS is not good fit for data.
4	In structured data, content is typically in text format.	In unstructured data, content is in the form of Text, Images, Audio, Video, Documents.
5	Representation is in discrete row and columns form.	Representation is less defined.
6	Data is stored in file format.	Data is in unmanaged file structure.
7	Example : Customer data Sales data	Example : Images Video

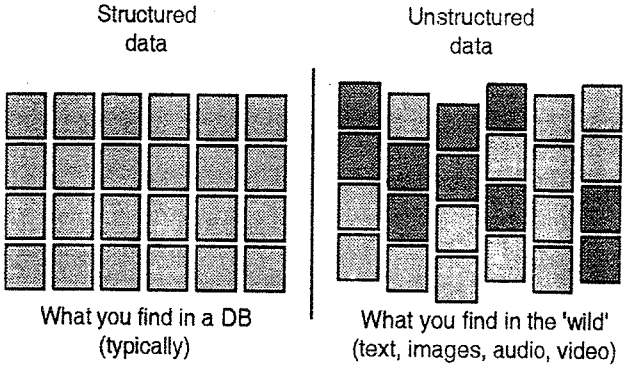


Fig. 6.4.1 : Structured and unstructured data

Syllabus Topic : Distributed Database Model

6.5 Distributed Database Model

A database system can be viewed as consisting of three software modules: a transaction manager (TM), a data manager (DM), and a scheduler.

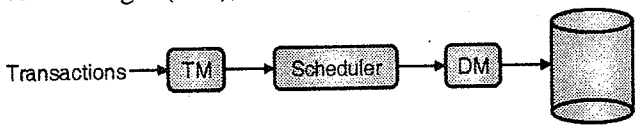


Fig. 6.5.1

- The transaction manager examines the execution of a transaction. It intercepts and executes all the transactions which have been submitted. Thus, the Transaction Manager is the mediator between users and the database system.
- Concurrency control is enforced by the scheduler. It grants or releases locks on data objects as per the requests of a transaction.
- The database is managed by the data manager. It performs the read-write requests issued by the transaction manager on behalf of a transaction by operating them on the database. The failure recovery is responsibility of data manager. Thus, the DM is the interface between the scheduler and the database.
- The concurrency control model of a distributed database system is shown in Fig. 6.5.2.

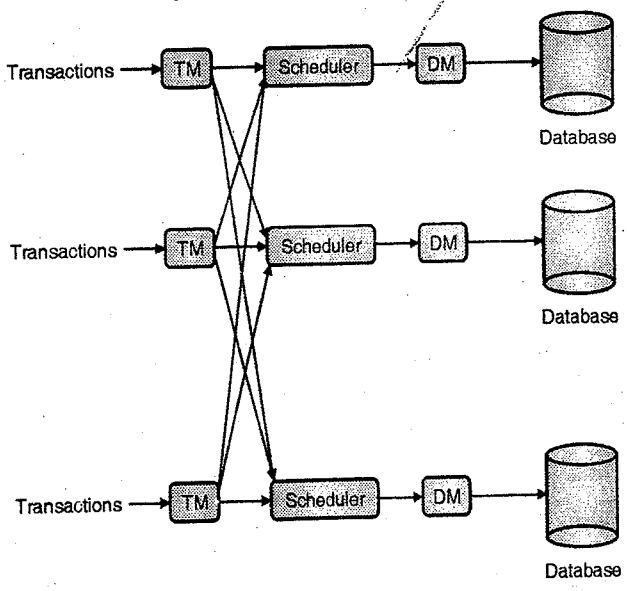


Fig. 6.5.2 : Distributed database system

Syllabus Topic : CAP Theorem and BASE Properties

6.6 CAP Theorem and BASE Properties

6.6.1 CAP Theorem

CAP stands for Consistency, Availability and Partitioning tolerance. In 2001 the CAP theorem was given by Eric Brewer, a professor at the University of California, Berkeley and one of the founders of Google, in the keynote of Principles of Distributed Computing.

Statement

- In general it is expected that every system should have Consistency, High-Availability and Partition-tolerance. But it is really hard for any system to achieve all these three properties at the same time.

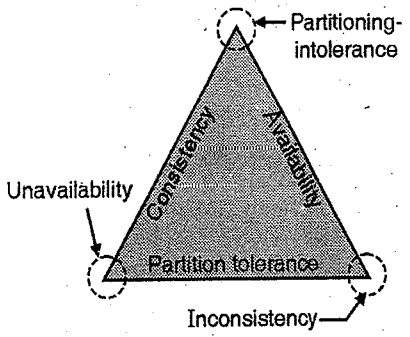


Fig. 6.6.1

- Maximum of two properties out of three can be achieved by the system. First we will see these three properties.

Consistency

- The system should follow the rule of sequence of updates which is present across all replicas in a cluster.
- Regardless of the location the data should be consistent. For example if client1 writes data in sequence A, B then another clients should be able to read the data in same order written by client1. It is also known as strong consistency.

Availability

- A service should be available to operate fully. It should be guaranteed that every request receives a

response about the status (successful or failed). It is also possible that a system which is not available may be consistent. It is hard to achieve *consistency* and *availability* at the same time.

- Both are contradictory to each other, i.e. if consistency is relaxed then system is highly available under the partitioning conditions (see next definition) and *strong consistency* means that in some specific conditions the system will not be available.

Partition Tolerance

- If failure may occur in any part of the system, the entire system does not get shut down.
- For example, consider cluster of n replicated nodes and network is unavailable among some number of nodes because of some reasons like network cable got chopped. Because of this synchronization of data is not possible.
- Here some part of system is in working condition while the other is not. If there is partition in the network, then there is possibility to lose consistency as we allow updates to both sides of the partition. Or we lose availability as we may shut down the system because of error until condition is resolved.
- A simple meaning of this theorem is "It is impossible for a protocol to guarantee both consistency and availability in a partition prone distributed system". This was mentioned above in example.

Adjustment between Requirements

1. **Available and Partition-Tolerant :** You have two nodes and the link between the two is severed. Since both nodes are up, the system can be designed to accept requests on each of the nodes. This insures the availability of system even if the network is partitioned. However independent results are issued by each node. Here high availability and partition tolerance is provided by compromising consistency.
2. **Consistent and Partition-Tolerant :** You have three nodes and one node loses its link with the other two. A rule cab made that a result will be returned only when a majority of nodes agree. So, even if the partition is there, the system will return a consistent result. Although the separated node is up, it won't be available since it is not able to reach consensus.
3. Finally, a system can be both *consistent and available*, but it possible that it may have to block on a partition. Most of the NoSQL database system architectures favour one factor over the other.

6.6.2 BASE Properties SPPU - May 16

University Question

Q. Explain BASE properties of NOSQL database with suitable example. **(May 2016, 5 Marks)**

- In the Chapter 4, we have seen the ACID properties of DBMS. ACID properties are an important concept for databases.
- Let's recall in brief what ACID means in traditional RDBMS community before moving to the BASE Properties.

The four properties are as follows.

- **Atomicity :** This property states that, either all operations contained by a transaction are done successfully or none of them complete at all.
- **Consistency :** The consistency property ensures that the transaction executed on the database system will bring the database from one valid state to another.
- **Isolation :** In case of concurrent transactions, the isolation property ensures that the system state should be same that would be obtained if transactions were executed sequentially.
- **Durability :** The durability property assures that after transaction committed successfully the updates made should remain permanent in the database even in the event of power loss, crashes, or errors.

ACID provides *strong consistency* (synchronous transactions) for partitioned databases and thus provides less availability.

Consistency is always preferable, but it is not always available.

- **Base** Property means **B**asically **A**vailable, **S**oft state, **E**ventual consistency.
- Consistency is always preferable, but it is not always available. In the NoSQL world, ACID transactions are less suitable as some DBMS do not have requirements for strict consistency, data freshness and accuracy in order to gain other benefits, like scale and resilience. By considering this, the BASE Consistency model is developed.

- The BASE properties are as follows :
  1. **Basic Availability** : Most of the time the database appears to work.
  2. **Soft-state :** It is not necessary that the stores should be write-consistent. Also the different replicas have to be mutually consistent all the time.
  3. **Eventual consistency :** Stores may show the consistency at some later point. Eventual consistency which is normally asynchronous in nature is a form of a weaker consistency which improves speed and availability.
- The BASE (Basically Available, Soft state, Eventual consistency) is the opposite of ACID.
- ACID is pessimistic i.e. consistency is required at the end of every operation. BASE is optimistic, i.e. it accepts that there is uncertainty in consistency.
- A BASE Model focus on availability as it is important for scale, but it doesn't offer guaranteed consistency of replicated data at write time.
- At the end we can say the BASE model of consistency provides a less strict assurance than **ACID :** Data will be consistent in the future, either at read time or may be always consistent for some points.

Syllabus Topic : Comparative Study of SQL and NoSQL

6.7 Comparative Study of SQL and NoSQL

For managing the database SQL has been most widely used programming language over the last few decades. It is a Relational Database Management System. But in today's era NoSQL has arises for an option to the SQL. There is very high difference between SQL and NoSQL. In this topic we will see the comparative study of SQL and NoSQL.

The conceptual difference is :

A framework of a relational database which is setup with the defined categories used by SQL. The tables of SQL are idle for storing data which is structured. The structured data like name, address fits into the SQL format perfectly.

But when data is unstructured it needed another format which is not dependant on the relationships of the data. When this situation occurs that time we can use NoSQL. NoSQL allows for the storage of an unstructured data without categorizing the data into fixed tables. NoSQL database scales horizontally. NoSQL is also known as Non-relational database or distributed database.

Factual difference is

- In SQL databases are structured in the form of tables, but in NoSQL databases are structured in the form of documents, graphs, or key-value pairs.
- In SQL Database there is a standard definition of schema which must be worked with the structured data. While In NoSQL there is no standard definition for schema which must be worked with the structured data.
- SQL database have predefined schema for structured data while NoSQL database have dynamic schema for unstructured data.
- SQL databases has feature of vertical scaling while NoSQL databases has feature of horizontal scaling.
- SQL database are designed and managed with SQL (Structured Query language) while NoSQL database are designed and managed with the UnQL (Unstructured Query Language).
- The syntax of SQL does not vary with database, while the syntax of UnQL varies with database.
- SQL is preferable for handling complex query while NoSQL cannot handle complex query. SQL queries are more powerful than NoSQL queries.
- Examples of SQL databases are : MySQL, Oracle, MS-SQL while examples of NoSQL are : MongoDB, BigTable, Redis, Hbase, Neo4i and CouchDb.
- SQL cannot manage big data which is stored in hierarchical manner while NoSQL handles hierarchical data better than SQL. Hence, NoSQL is preferable to SQL when managing big data.

Comparision between SQL and NoSQL

Sr. No.	SQL	NoSQL
1.	SQL means structured query language.	NoSQL means NOT only SQL language.
2.	Data is in structured and organized form.	Data is in unstructured form.
3.	It is used for small to medium scale data set effectively.	It is used for large set of data.
4.	SQL is schema based.	NoSQL is schema less.
5.	Data is stored as row and column in table where each column is of specific type.	The data model is depending upon the database type.
6.	Relational database is table based.	Key value pair, storage, column, document store, graph database.
7.	Example : SQL server Oracle	Example : MongoDB HBase

Syllabus Topic : NoSQL Data Models

6.8 NoSQL Data Models

Data Model

Data model is the mode which organizes the data. It is a representation which is used to understand and manipulate the data. The data model helps to :

- Represent the data elements in analytical view.
- Maintain relationship of these elements.
- Describe the way by which we interact with the database.

In RDBMS the relational model was used to represent the data. The data is represented in the form of tables (combination of rows and columns). Each row represents some entity while columns represent relationships in the same table or with another table. The relational data modeling has been a well-defined discipline for many years. However now a days with the emergence of NoSQL databases, need of a new data model arises.

NoSQL Data Model

The NoSQL data model is different from traditional relational data model. There are different data models :

- Key-value
- Document
- Column - family
- Graph

The three model Key-value, document and column- family share common features of Aggregate Orientation.

NoSQL Agreegate Model

- In this model it considered that we have to manage more complex data compared to relational model. The complex data structure are in the form of map, list etc. The aggregate model uses this complex structure. Aggregate is a collection of data objects which are treated as a single unit to manage and manipulate. Atomic operations are expected to update aggregates. Using aggregate it is easy to work on cluster which is unit of machines. Aggregates helps application developer by solving the mismatch problem which usually occur in relational model.
- When the aggregate model runs on a cluster, it gives several advantages on computation power and data distribution. While gathering data, it requires minimizing the number of nodes. The aggregate gives an important view that which data should be stored together.
- The **Key-Value** and **Document** databases are strongly aggregate oriented. These databases contain number of aggregates with a key to get the data. In Key-Value we can store any type of object.
- The Column-Family has two level aggregate structure. The first key is the row identifier. The second level values are defined to as columns.

Important point related to NoSQL Data Aggregate Model

- All these models use aggregated index by a key.
- The key is used for searching the data.
- The Aggregate acts as a atomic unit for modification.

- In the document model, the document is treated as single unit of storage. This model makes document transparent for querying.
- In Column-Family model, the columns are divided into column families and treated as single units.
- All models improve the accessibility of data.

Syllabus Topic : Case Study - Unstructured Data from Social Media

Case study is in detail study of a phenomenon, like a person, group, or situation. The phenomenon is studied in detail, cases are analyzed and solutions or interpretations are presented.

6.9 Case Study - Unstructured Data from Social Media

In this case study we are going through following steps.

1. Background
2. Problem
3. Proposed Solution
4. Implementation
5. Results

1. Background

- The unstructured data is the form of information which does not have any predefined data model. This data is not suitable for traditional relational model. It is usually text-heavy, but also contains data like as dates, numbers etc.
- Such data is difficult to understand and manage using traditional programs because it generates irregularities and ambiguities.
- In social media maximum of data is in unstructured form. The most big data sources like WhatApp, Facebook, twitter have unstructured data. It is difficult to run any analytics on such data. To analyze such data we need to convert it into structured form.

2. Problem

For social media sites, it is really hard to organize such data. There are various problems to handle the data which lead to find out some solution. Social media contains different types of data some of which is important and some not.

- Chat History
- Posts of users
- Free advertisements posted by users
- Twits
- Comments

There is no any predefined or standard format of this data. Tweets, Facebook postings and other social comments have to be analyzed to determine the sentiment of the population. The business strategies are decided on this data.

3. Proposed Solution

To handle the unstructured data properly it should be converted into structured or semi structured format. But if volume of the data is very big then it is difficult to convert it. For such big data we can use the database framework like NoSQL and Software system like Hadoop.

There are two solutions for handling unstructured data :

A. Convert it into structured format.

B. Use database framework like NoSQL and Software system like Hadoop.

4. Implementation

A. Converting unstructured data into structured format

The unstructured data can be converted into semi-structured or structured data by converting the text into words and phrases which can fit into relational tables. Then it can be categorized. The analysis techniques can then be used to conclude and arrive at results. It is important that the type of data, the source and its general understanding has to be re-vamped. So does this process of converting unstructured to structured data may be manual or machine driven through algorithms. Algorithms reduce accuracy but increase scale.

B. Use database framework like NoSQL and Software system like Hadoop.

We have already seen the database framework NoSQL which can deal with such unstructured data. The database tools like Hadoop, MongoDB also are useful in dealing with unstructured data.

5. **Results**

Both above options are 100 percent effective in dealing with the vast social media unstructured data. Both options give a perfect solution to manage this data. Data management involves access and manipulation of data.

Syllabus Topic : Introduction to Big Data

6.10 Introduction to Big Data

Now a day the amount of data created by various advanced technologies like Social networking sites, E-commerce etc. is very large. It is really difficult to store such huge data by using the traditional data storage facilities.

Until 2003, the size of data produced was 5 billion gigabytes. If this data is stored in the form of disks it may fill an entire football field. In 2011, the same amount of data was created in every two days and in 2013 it was created in every ten minutes. This is really tremendous rate.

In this topic, we will discuss about big data on a fundamental level and define common concepts related to big data. We will also see in deep about some of the processes and technologies currently being used in this field.

Big Data

Big data means huge amount of data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big Data is complex and difficult to store, maintain or access in regular file system. Big Data becomes a complete subject, which involves different techniques, tools, and frameworks.

There are various sources of big data. Now a days in number of fields such huge data get created. Following are the some of fields.

- **Stock Exchange :** The data in the share market regarding information about prices and status details of shares of thousands of companies is very huge.
- **Social Media Data :** The data of social networking sites contains information about all the account holders, their posts, chat history, advertisements etc. On topmost sites like facebook and whatsapp, there are literally billions of users.
- **Video sharing portals :** Video sharing portals like youtube, Vimeo etc. contains millions of videos each of which require lot much of memory to store.
- **Search Engine Data :** The search engines like Google and Yahoo holds lot much of metadata regarding various sites.
- **Transport Data :** Transport data contains information about model, capacity, distance and availability of various vehicles.
- **Banking Data :** The big giants in banking domain like SBI or ICICI hold large amount of data regarding huge transactions of account holders.
- The data can be categorized in three types.
  1. **Structured Data :** This type of data is stored in relations(tables) in Relational Database Management System.
  2. **Semi-structured Data :** This type of data is neither raw data nor typed data in a conventional database system. A lot of data found on the web can be described as semi-structured data. This type of data do not have any standard formal model. This data is stored using various formats like XML and JSON.
  3. **Unstructured Data :** This is data do not have any pre-defined data model. The data of video, audio, Image, text, web logs, system logs etc. comes under this category.
- In general there are some important issues regarding data in traditional file storage system.
  1. **Volume :** Now a days the volume of data regarding different fields is high and potentially increasing day by day. Organizations collect data from a variety of sources, including business transactions, social media and information etc.
  2. **Velocity :** The configuration of system single processor, limited RAM and limited storage capacity system cannot store and manage high volume of data.
  3. **Variety :** The form of data from different sources is different.
  4. **Variability :** The flow of data coming from sources like social media is inconsistent

because of daily emerging new trends. It can show sudden increase in size of data which is difficult to manage.

5. **Complexity :** As the data is coming from various sources, it is difficult to link, match and transform such data across systems. It is necessary to connect and correlate relationships, hierarchies and multiple data linkages of the data.

- All these issues are solved by the new advanced **Big Data Technology.**

Big Data Technologies

- Big data technologies are based on the accuracy of analyzing the data. This leads to more operational efficiencies, reductions in cost, reduction in failure and data loss.
- To implement the Big Data Technology, there is requirement of strong infrastructure which can manage and process the big data. This data may be structured or unstructured. The infrastructure must be able to protect data privacy and security.

Characteristics of big data

- Big data stores and manages huge amount of data in time and cost effective manner.
- It can analyze data of any form like unstructured, structured or streaming.
- It can maintain multiple copies of the important data across clusters.
- The System Failure Mechanism of this technology is very strong which avoid the data loss.
- Data can be stored in blocks on different machines which can be merged any times as per requirement.
- It captures data from live events in real time.
- There are various technologies in the market from different vendors including Amazon, IBM, Microsoft, etc., to handle big data. There are two classes of Big Data technology which can handle big data.
  - Operational Big Data
  - Analytical Big Data

Operational Big Data

- It was developed to address the shortcoming of traditional database and it is faster and can deal with large quantity of data spread over multiple servers. We are also using cloud computing architectures to allow massive computation to run effectively as well as it is cost efficient. This has made Big Data workload easier to manage, faster to implement as well as cheaper. Here in addition to interaction with user it also provides artificial intelligence about the active data. For example in games the moves of user are studies and next course of actions are suggested.
- Systems like MongoDB and NoSQL comes under this category.
  - **MongoDB :** This system provides operational capabilities for interactive, real-time workloads where data is primarily captured and stored.
  - **NoSQL :** The new advanced cloud technology is used to implement the huge computations to be run efficiently and inexpensively. It helps to manage the data easily and fast. This Cloud Technology is used in NoSQL big data system.

Analytical Big Data

- These systems provides analytical capabilities for complex analysis which considers most or all of the data.
- For example Massively Parallel Processing (MPP) database systems and MapReduce.
- Analytical Big Data is addressed by MPP database systems and MapReduce. These technologies has evolved as a result of shortcoming in traditional database which deals with one servers only. On the other hand MapReduce provides new method of analyzing data which is beyond the scope of SQL.

Challenges regarding Big Data

There are various challenges relate to big data :

- Getting the source data
- Making correction in this data
- Storing the data
- Searching specific data depending upon some criteria
- Sharing data between machines and users
- Transferring data
- Analyzing the data

Syllabus Topic : HADOOP - HDFS, MapReduce

6.11 Hadoop

SPPU - Dec. 15

University Question

Q. What is Hadoop? (Dec. 2015, 2 Marks)

- **Hadoop** is an open source, Java-based programming framework which supports the processing and storage of extremely large sets of data in a distributed computing environment using simple programming models.
- Hadoop has very strong processing power and the ability to handle virtually unlimited parallel tasks.
- With the help of Hadoop, applications can be run on systems with thousands of commodity hardware nodes. It can handle thousands of terabytes of data. Hadoop has distributed file system which facilitates rapid data transfer rates among nodes. This allows the system to proceed even in case one or more nodes get failed. This approach avoids the unexpected data loss.
- Hadoop has quickly emerged as a foundation for big data processing tasks like scientific analytics of data, planning of business and sales, and processing enormous volumes of data including social media data.

History of Hadoop

- Computer scientists Doug Cutting and Mike Cafarella created Hadoop in 2006 to support distribution for the Nutch (search engine). The main aim is to increase the speed of search results by the distribution of data and implement calculations on different computers by multitasking.
- Later on Cutting joined Yahoo but him still works on the Nutch project with the ideas based on Google's early work with automating distributed data storage and processing.
- The Nutch Prject divided into two parts –
  - Nutch - Web crawler portion
  - Hadoop - Distributed computing and processing portion
- In 2008, Yahoo released Hadoop as an open-source project. Now Apache Software Foundation (ASF) manages the Hadoop's framework and ecosystem of technologies.

6.11.1 Modules (Components) of Hadoop

SPPU - Dec. 14, Dec. 15, May 16, Dec. 16

University Questions

Q. Explain different components of HADOOP. (Dec. 2014, Dec. 2015 7 Marks)

Q. Explain in brief different building blocks of HADOOP. (May 2016, Dec. 2016, 7 Marks)

- **HDFS : HDFS** stands for Hadoop Distributed File System. It states that the files will be broken into blocks and stored in nodes over the distributed architecture. It provides high-throughput access to application data.

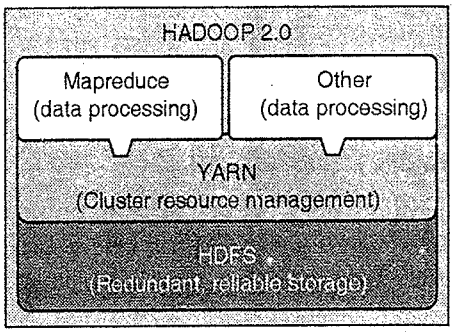


Fig. 6.11.1

- **Yarn :** Yarn stands for "Yet another Resource Negotiator". It is used for job scheduling and managing the cluster (multiple nodes).
- **Map Reduce :** This is YARN-based system for parallel processing of large data set using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair.
- **Hadoop Common :** These Java libraries and utilities are used to start Hadoop. These are used by other Hadoop modules. These libraries provide file system and OS level abstractions.

Advantages of Hadoop

1. Huge amounts of any kind of data can be stored and processed quickly.
2. **Computing power :** Hadoop's distributed computing model processes big data fast.
3. **Fault Tolerance :** In case of failure of any node the tasks are automatically redirected to other nodes.

4. **Flexibility :** Any kind of unstructured data like text, images and videos can be stored.
5. **Low cost :** This open-source framework is free.
6. **Scalability :** New nodes can be easily added to handle big tasks.

Disadvantage of Hadoop

1. Hadoop is rough in manner because the software is under active development.
2. Programming model is very restrictive.
3. Joins of multiple datasets are tricky and slow.
4. Cluster management is hard : In the cluster, operations like debugging, distributing. software, collection logs etc are too hard.
5. Requires care and may limit scaling.

6.11.1.1 MapReduce SPPU - May 15, May 16

University Question

Q. Write a short note on : MapReduce in Hadoop. **(May 2015, May 2016, 5 Marks)**

- MapReduce is an important part of Hadoop. It is a software framework which is used to write applications easily to process huge amount of data (multi-terabyte data-sets) simultaneously on large clusters (thousands of nodes) in reliable, fault-tolerant manner.
- MapReduce basically referes to two tasks performed by the Hadoop programs. One is map and another is reduce.

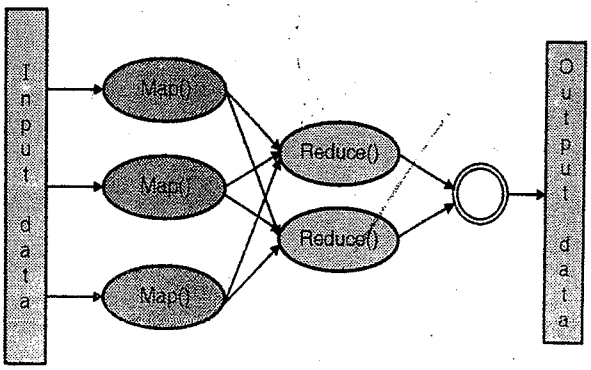


Fig. 6.11.2

- Hadoop programs perform following two tasks on **MapReduce** :
  - **The Map Task :** This is the first task, which takes a set of data and converts it into another set of data in which individual elements are broken down into tuples (key/value pairs).
  - **The reduce :** This job takes the output of previously executed map task as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.
- In MapReduce framework, the data of input and output is stored in a file system. The framework handles the scheduling of all the tasks, monitoring these tasks and if fails, re-executes them.
- The main advantage of MapReduce is that it is simple to scale data processing over multiple computing nodes. The data processing primitives are known as mappers and reducers in the MapReduce model.
- The MapReduce framework consists of single master JobTracker and one slave TaskTracker per cluster-node.
- **Master JobTracker :** The tasks under master are -
  - Managing the resources.
  - Tracking consumption and availability of resources.
  - Scheduling the jobs component tasks on the slaves.
  - Monitoring the tasks and re-executing the failed tasks.
- **The slaves TaskTracker :** It execute the tasks as per the directions of the master and provide task-status information to the master periodically.
- The JobTracker is very important in Hadoop MapReduce service. If JobTracker goes down, all running tasks get halted.

Advantages of MapReduce

1. Scalable.
2. Fault tolerant.
3. Simple coding model.
4. Supports unstructured data.

6.11.1.2 Hadoop Distributed File System (HDFS)

- The HDFS is the primary storage system used by Hadoop applications. HDFS is a distributed file system and a framework provided by Hadoop for the analysis and transformation of huge data sets which uses the MapReduce paradigm. The HDFS is based on Google File System (GFS). It provides

high-performance access to data across Hadoop clusters (thousands of computers), HDFS has become a key tool for managing pools of big data and supporting big data analytics applications.

- HDFS is usually deployed on commodity hardware of low-cost where the possibility of server failures is common. The file system is designed to be highly fault-tolerant. The HDFS facilitates the rapid transfer of data between different computer nodes and enables Hadoop systems to proceed its execution even if one or more nodes get failed. That decreases the risk of catastrophic failure, even in the event that numerous nodes fail.

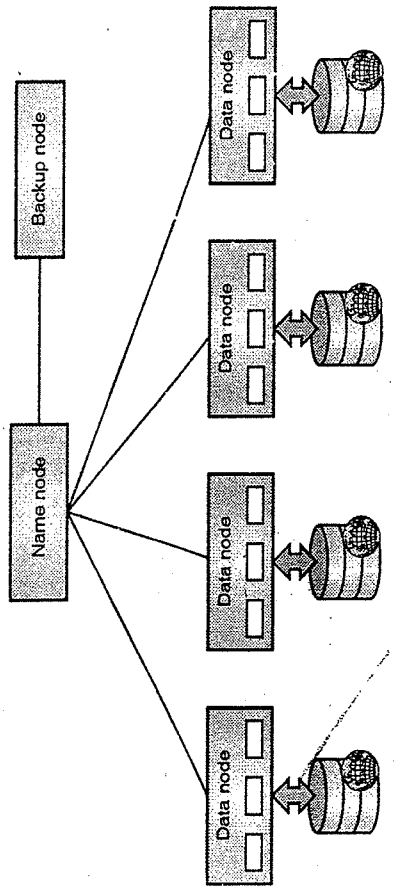


Fig. 6.11.3

- The architecture used by HDFS is known as master/slave architecture.
- NameNode which manages the metadata of file system and DataNode which stores the actual data.
- The HDFS namespace is a hierarchy of files and directories. Inodes are used to represent these file and directories. Inodes are used to record attributes such as permissions, modification and access times etc. The file content is split into large blocks and each block of the file is independently replicated at multiple DataNodes.
- The tree structure of namespace is maintained by the NameNode. It maps the blocks to DataNodes. In a cluster there may be hundreds of DataNodes and thousands of HDFS clients per cluster, as number of application tasks can be executed by each DataNode simultaneously.

Advantages of HDFS

1. High scalability.
2. Low limitation.
3. Open source.
4. Low cost.

Disadvantages of HDFS

1. Still rough - means software under active development.
2. Programming model is very restrictive.
3. Cluster management is high.

□□□

Note

Lab Manual

Lab Manual

Group A : Database Programming Languages - SQL, PL/SQL

Assignment 1 : Study of Open Source Relational Databases : MySQL.

Ans. :

What is MySQL?

- The world's most widely used open source relational database management system is MySQL. The name MySQL is a combination of "**My**", the name of co-founder **Michael Widenius' daughter**, and "**SQL**", the abbreviation for **Structured Query Language.**
- MySQL was owned and sponsored by a single **for-profit firm**, the Swedish company MySQL AB, now owned by **Oracle Corporation.**
- MySQL is written in **C and C++.** Its SQL parser is written in **yacc.**
- MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses.

Features

Several features are Available in MySQL 5.6(announced in February 2013) some of them are as follows :

- Has A broad subset of ANSI SQL 99, as well as extensions.
- Cross-platform support.
- Allows nested SELECT statements.
- Also supports Triggers, Cursors, Stored procedures, using a procedural language.
- Allows views modification.
- Embedded database library .
- Supports Commit grouping: commit grouping means gathering multiple transactions from multiple connections together to increase the number of commits per second.
- Built-in replication support (i.e., master-master replication and master-slave replication) with one master per slave, many slaves per master. Multi-master replication is provided in MySQL Cluster, and multi-master support can be added to unclustered configurations using Galera Cluster.

Command Line Interfaces

A Command Line Interface allows users to give commands to computer program by typing successive lines of text (command lines). For example, mysql client is a command line tool.

Graphical User Interfaces

A Graphical User Interface (GUI) allows users to interact with electronic devices or programs through graphical icons (such as buttons, menus). GUIs are easier to learn than command-line interfaces (CLIs), which require commands to be typed on the keyboard.

Several front ends are available that integrate with MySQL and enable users to work with database structure and data visually. Some well-known front ends are :

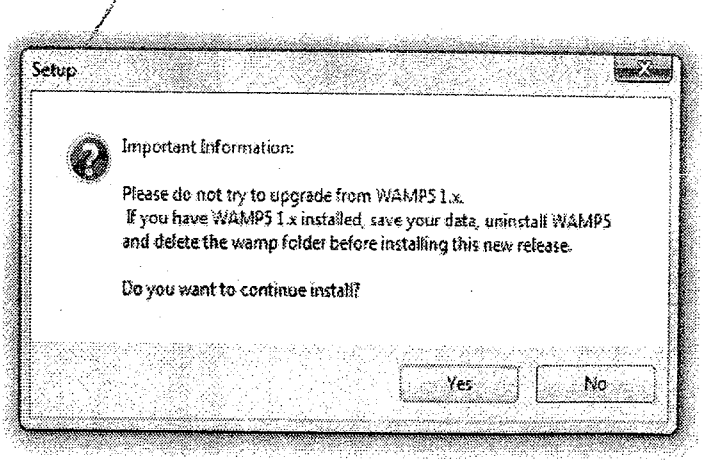
- **MySQL Workbench :** It was developed by MySQL AB, and enables users to graphically administer MySQL databases and visually design database structures.
- **Adminer :** Adminer (formerly known as phpMinAdmin) is a free MySQL front end for managing content in MySQL databases. Its author is Jakub Vrána who started to develop this tool as a light-weight alternative to phpMyAdmin, in July 2007.
- **Navicat :** Navicat is a series of graphical database management and development software produced by PremiumSoft CyberTech Ltd. for MySQL, MariaDB, Oracle, SQLite, PostgreSQL and Microsoft SQL Server. Navicat is a cross-platform tool and works on Microsoft Windows, OS X and Linux platforms. Upon purchase, users are able to select a language for the software from eight available languages: English, French, German, Spanish, Japanese, Polish, Simplified Chinese and Traditional Chinese.
- **phpMyAdmin :** PhpMyAdmin is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser. It can perform various tasks such as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions. The software, which is available in 78 languages, is maintained by *The phpMyAdmin Project.*
- **SQLBuddy :** SQLBuddy is an open-source web-based application written in PHP intended to handle the administration of MySQL and SQLite with the use of a Web browser. The project places an emphasis on ease of installation and a simple user interface
- **SQLyog :** Data manipulations (e.g., insert, update, and delete) may be done from a spreadsheet-like interface. Its editor has syntax highlighting and various automatic formatting options.

Setting up Environment for using MySQL

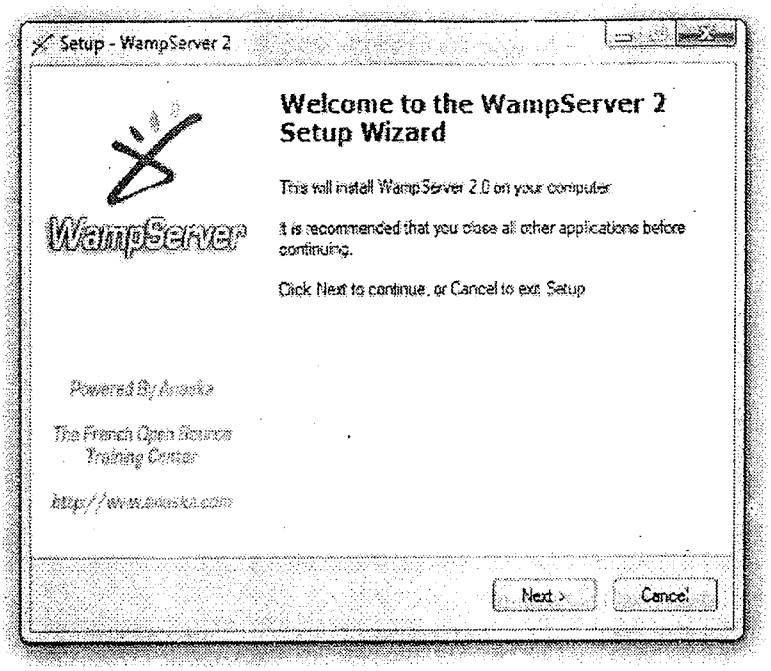
WAMP provides the popular combination of the PHP server-side language and the MySQL database in one "easy-to-install" package. So, Let's install WAMP server as follows:

WAMP Installation

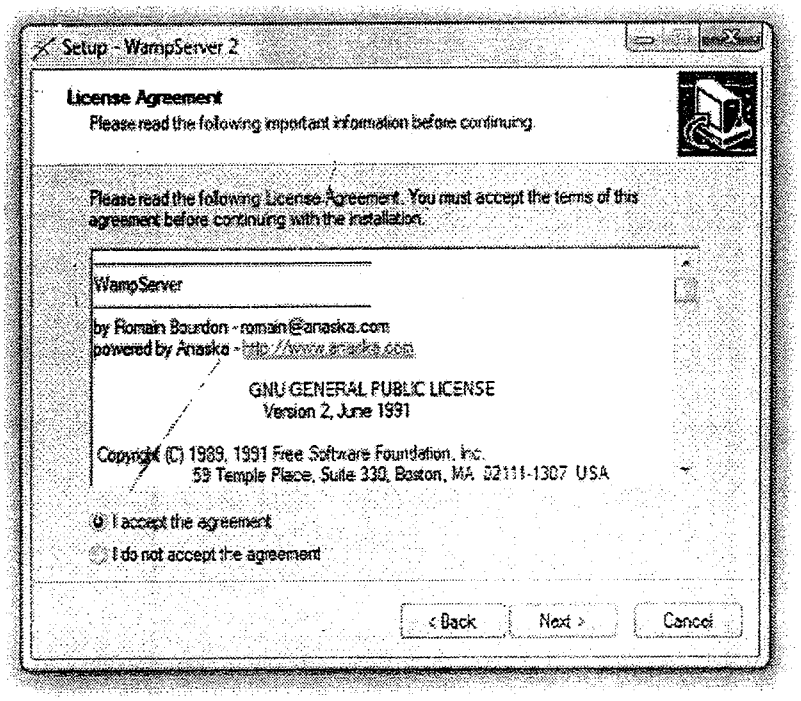
- Download latest release of wampserver2 from http://www.wampserver.com/en/.
- Once download double-click the file to launch the installer.
- The setup will show you the warning regarding previous version installed. Uninstall it if it exists otherwise continue the setup.



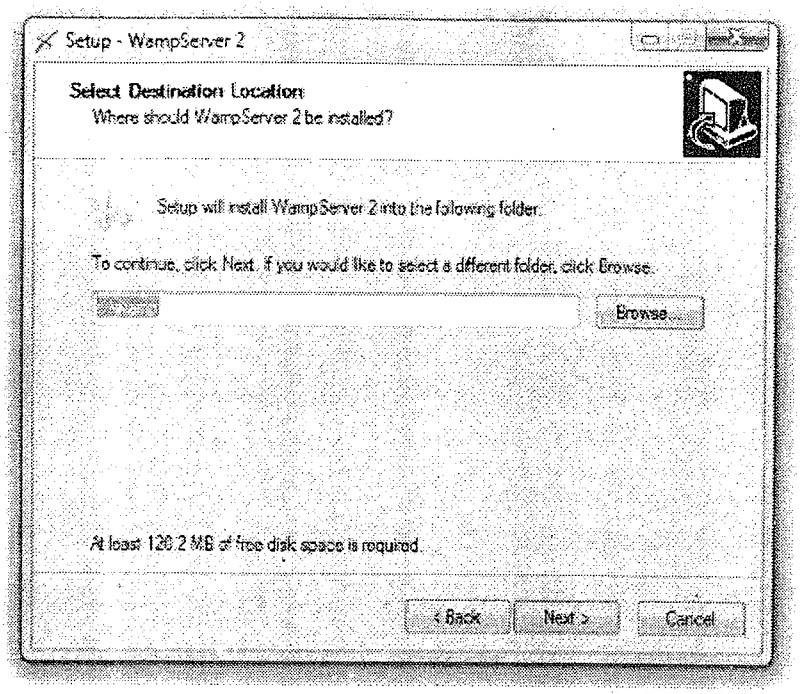
- Click on **Next**.



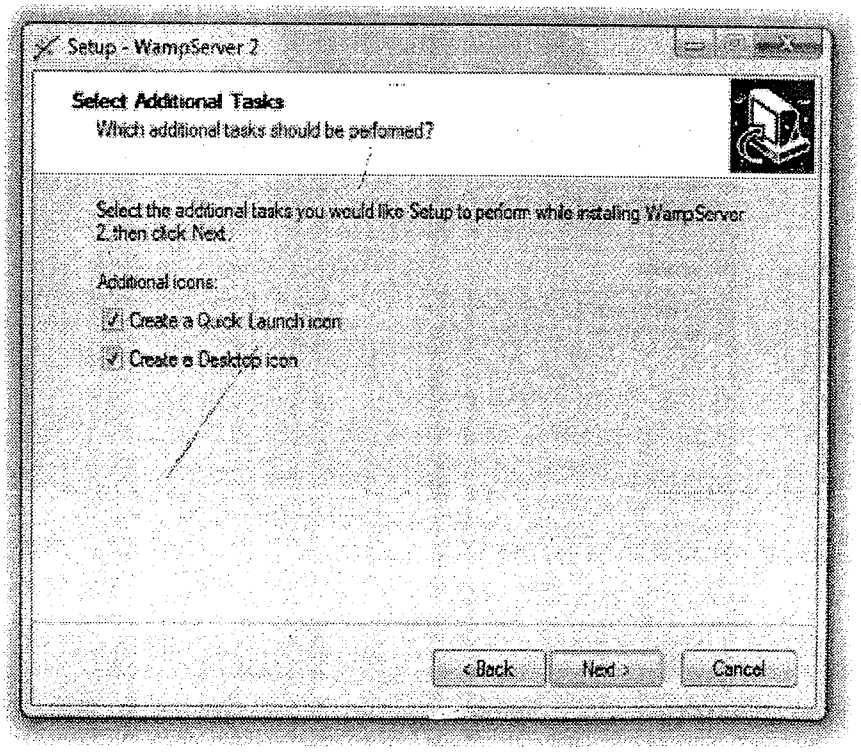
- Review the **License Agreement**, select **I accept the agreement** and then click the **Next** button.



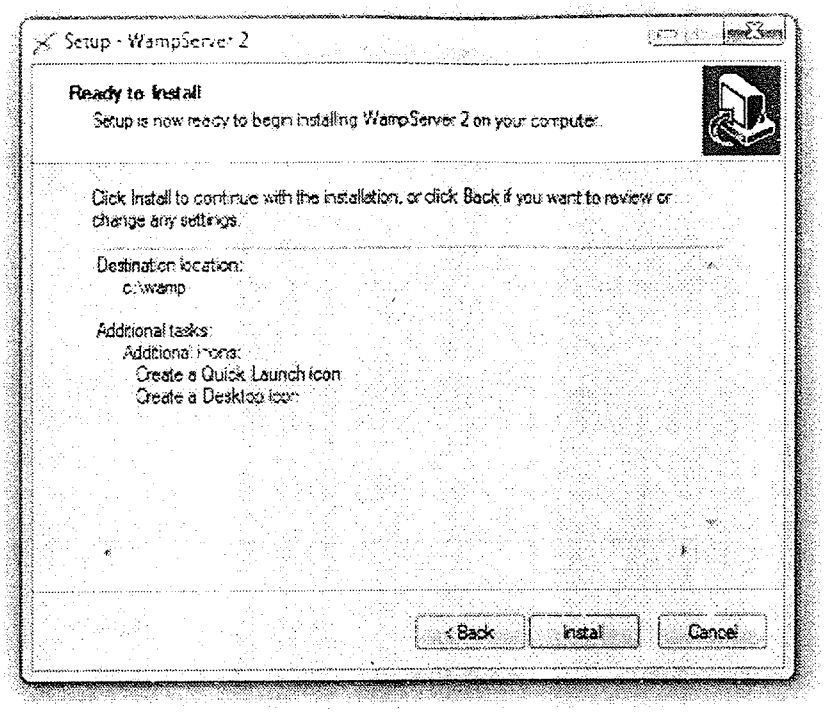
- It's a good idea to keep the default directory – "c:\wamp" for installation. Otherwise you can change the path.



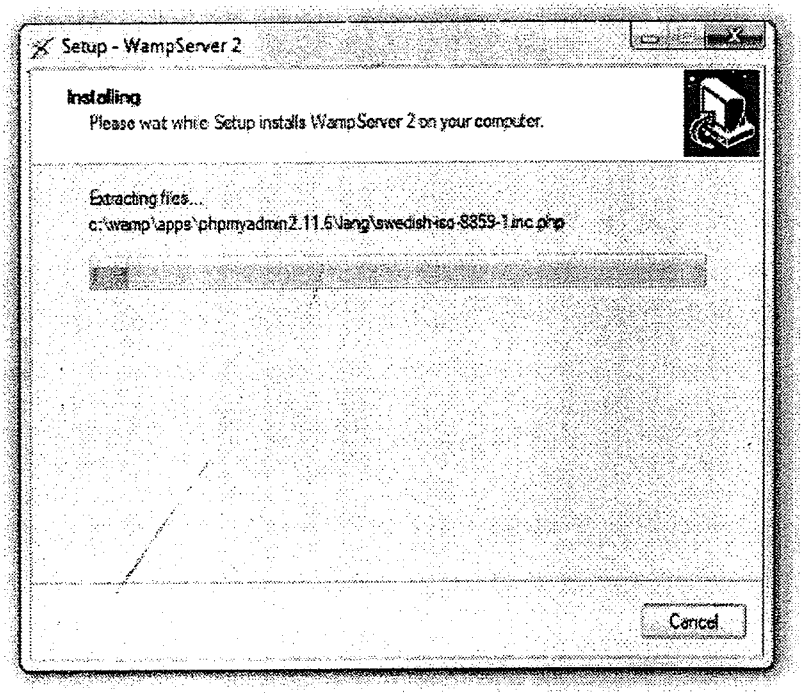
- Decide if you want to have WAMP Quick Launch and/or Desktop icons, and click **Next.**



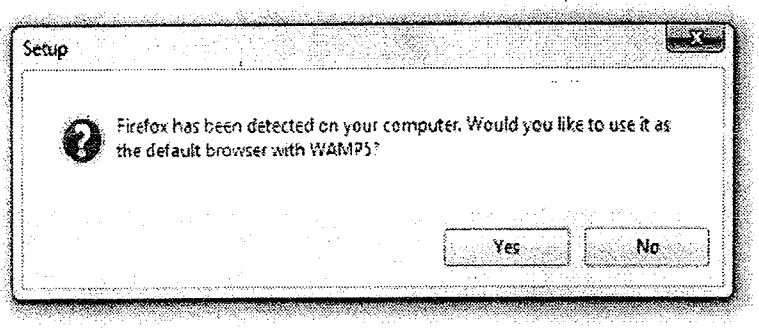
- Finally, click **Install**.



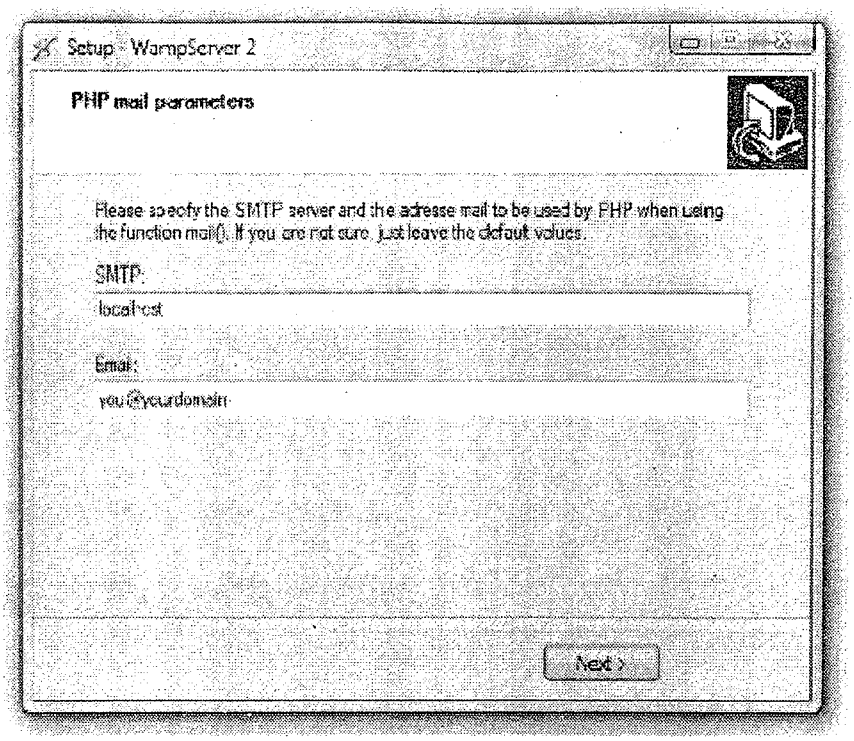
- The installation doesn't take too long.



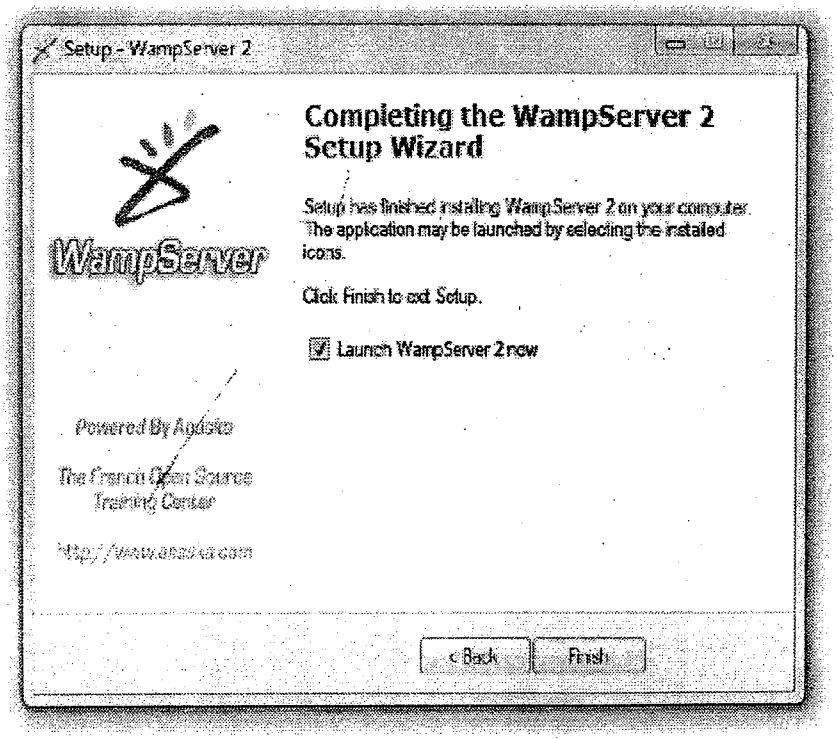
- After it complete. Following window will occur. To ask you about browser you want to use as your default browser. Click **Yes**.



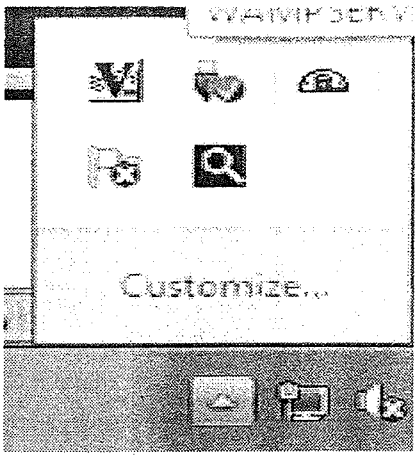
- Leave the **SMTP :** Server set as **localhost** (more on this later) but **do** change the email address to yours. Click **Next**.

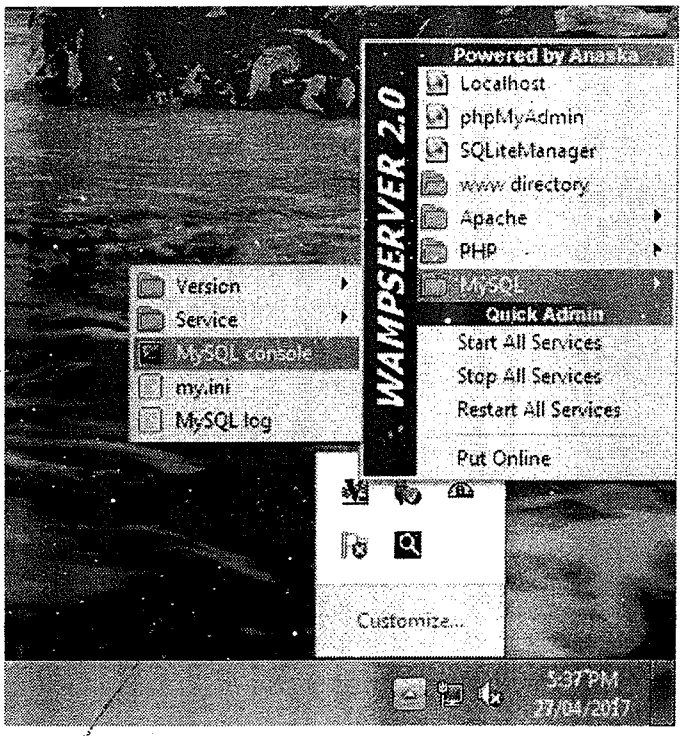


- Select the **Launch WampServer 2 now** checkbox and click **Finish**.

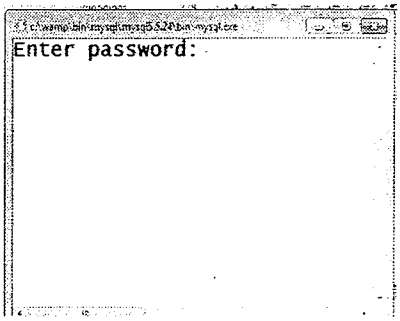


- After the installation you should have a fully functional Apache web server with PHP and MySQL.
- To launch your web server go to *Start › All Programs › WampServer* (or wherever you installed WAMP to) and click start WampServer.
- You'll notice a small icon in the bottom right corner of your screen: click on that icon and select MySQL>MySQL Console.





- The console ask for MySQL password, by default the password is null so just press ENTER key of keyboard. You can change the password later.



- **Basic Queries :** Once logged in, you can try some simple queries. Before starting to execute the queries first we will know some basic rules to write queries as follows:

Note : 1. Most MySQL commands end with a semicolon(;)
2. After every query, MySQL returns the total number of rows found, and the total time to execute the query.

- Keywords may be entered in uppercase or lowercase as it is not case sensitive. The following commands are equivalent:

```
mysql>select current_date();
mysql>select CURRENT_DATE();
```

- You can also enter multiple statements on a single line. Just end each one with a semicolon.

1. To display the MySQL version, type SELECT VERSION( )

Syntax

```
Mysql> SELECT VERSION();
```

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.5.24-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statemen

mysql> select version();
+------------+
| version()  |
+------------+
| 5.5.24-log |
+------------+
1 row in set (0.00 sec)

mysql>
```

2. To display current date

```
mysql>select CURRENT_DATE( );
```

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.24-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

mysql> select current_date();
+----------------+
| current_date() |
+----------------+
| 2017-05-31     |
+----------------+
1 row in set (0.00 sec)

mysql>
```

3. MySQL can be used as a simple calculator

```
mysql> select 4+5;
+-----+
| 4+5 |
+-----+
|   9 |
+-----+
1 row in set (0.00 sec)

mysql> select pi()/2;
+--------------+
| pi()/2       |
+--------------+
| 1.5707963268 |
+--------------+
1 row in set (0.00 sec)
```

4. To exit the MySQL Shell, just type QUIT or EXIT

```
Mysql> QUIT
Or
Mysql> EXIT
```

- As we use MySQL first time, we need to create a database to perform DML and DDL queries on that.

1. Create database using CREATE DATABASE <database name>; command.

- To check currently present databases in the system use **SHOW DATABASES;** command.

```
mysql> create database userdb;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db                 |
| mysql              |
| person             |
| user               |
| userdb             |
+--------------------+
6 rows in set (0.03 sec)

mysql>
```

- To execute SQL queries in MySQL we need to choose a database so after creating database use it. To use the created database type following command on MySQL console: **USE <database name>;**

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.5.24-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use userdb;
Database changed
mysql>
```

- Now you can execute DDL and DML queries on "userdb" database.

Assignment 2 : Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym.

Ans. :

SQL commands can be divided into three main types

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)

Among this only DDL commands are described in this assignment remaining commands will be described later.

1. Data Definition Language (DDL)

Data definition language contains the commands which are used to create and destroy the databases and their objects (like, table, view, index, etc.). Following commands are come under DDL.

- To create the database instance - **CREATE**
- To alter the structure of database - **ALTER**
- To rename database instances - **RENAME**
- To drop database instances - **DROP**

DDL commands on database object : Table

A) Creating Table

The **CREATE TABLE** statement is used to create table in database.

Syntax

```
CREATE TABLE table_name (
   column1 datatype[size],
   column2 datatype [size],
   column3 datatype[size],
   ....
);
```

Example

Following command creates a table Employee having four columns : Employee_no, Employee_name, Joining _date, and Salary.

```
CREATE Table Employee (
Employee_no integer(3),
Employee_name varchar(20),
 joining_date date ,
Salary integer(6)
);
```

Creating New Table from Existing Table

Consider the existing table Employee

Creating new table same as of existing table

Syntax

```
Create table table_name as select * from existing_table_name;
```

For Example

```
Create table newEmployee1
```

As

```
select * from Employee;
```

The newly created table newEmployee1 will include all the fields and records as in Employee table.

Creating new table having specific fields but all the records from existing table

Syntax

```
Create table table_name as select field_1,field_2... from existing_table_name;
```

For Example

```
Create table newEmployee2
```

As

```
select Employee_no, Employee_name from Employee;
```

After executing this command the newly created table will contain two fields such as Employee_no and Employee_name, and also contain all the corresponding records as in Employee table.

Creating new table having specific records but all the fields from existing table

Syntax

```
Create table table_name as select * from existing_table_name where condition
```

For Example

```
Create table newEmployee3
```

As

```
select * from Employee
where Salary > 80000;
```

The newly created table will have same structure as of employee table, but it will contain records of only those Employees who got Salary above 80000 Rs.

Creating new table having no records but all the fields from existing table

That means copying only structure of existing table.

Syntax

```
Create table table_name as select * from existing_table_name where false condition
```

For Example

```
Create table newEmployee4
```

As

```
select * from Employee
where 1=2;
```

Here, 1=2 is the false condition

The newly created table will have same structure as of existing table, but it will not copy any records from.

B) Modifying Table

ALTER TABLE query is used to modify structure of a table which is already exists in the database. We can add, delete or modify column.

1. Adding new column in a table

Syntax

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

For Example

```
mysql> CREATE Table Employee (
    -> Employee_no integer(3),
    -> Employee_name varchar(20),
    ->  joining_date date ,
    -> Salary integer(6)
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> ALTER TABLE  Employee ADD Department  varchar(15);
Query OK, 0 rows affected (0.23 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

2. Dropping column from table

Syntax

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

For Example

```
mysql> ALTER TABLE  Employee DROP column Department;
Query OK, 0 rows affected (0.17 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

3. Modifying Column of a Table

Here we are changing the data type and size of column roll_number.

Syntax

```
ALTER TABLE table_name MODIFY COLUMN column_name data_type;
```

For Example

```
mysql> ALTER TABLE Employee MODIFY COLUMN  Employee_no  varchar(4);
Query OK, 0 rows affected (0.17 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

C) Renaming Table

Syntax

```
rename table current_table_name to new_table_name;
```

For Example

```
mysql>  rename table Employee to Emptable;
Query OK, 0 rows affected (0.06 sec)

mysql>
```

D) Drop Table

DROP TABLE query is used to delete table.

Syntax

```
drop table table_name;
```

For Example

```
mysql> drop table  Emptable;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

DDL Commands on Views

In SQL, a view is a virtual table containing the records of one or more tables based on SQL statement executed. Just like a real table, view contains rows and columns. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table. The changes made in a table get automatically reflected in original table and vice versa.

A) Creating View

Consider we have existing table as Employee (Employee_no, Employee_name, joining_date, Salary).

1. Creating view having all records and fields from existing table

Syntax

```
CREATE or replace VIEW view_name
AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

For Example

```
Create or replace view Emp_view1
```

As

```
select * from Employee;
```

This statement will create view having all the fields as in Employee table.

2. Creating view having specific fields but all the records from existing table

Syntax

```
Create or replace view view_name
```

As

```
select field_1, field_2... from existing_table_name;
```

For Example

```
Create or replace view Emp_view2
```

As

```
select Employee_no, Employee_name from Employee;
```

This statement will create view having specific fields from Employee table.

3. Creating new view having specific records but all the fields from existing table

Syntax

```
Create or replace view view_name
```

As

```
select * from existing_table_name
where condition;
```

For Example

```
Create or replace view Emp_view3
```

As

```
select * from Employee
where Salary > 80000;
```

B) Updating View

Update query is used to update the records of view. Updation in view reflects the original table also. Means the same changes will be made in the original table also.

Syntax

```
UPDATE view_name
set field_name = new_value;
where condition;
```

For Example

```
UPDATE Emp_view1
set Salary = 73000
where Employee_no=102;
```

C) Dropping View

DROP query is used to delete a view.

Syntax

```
DROP view view_name;
```

For Example

```
DROP view Emp_view2;
```

DDL Commands on Index

An index is a pointer to data in a table. An index in a database is similar to the alphabetical index of a book present at the end of book.

Indexes can be created or dropped with no effect on the data.

A) Creating Index

CREATE INDEX statement is used to create an index. In this statement we have to mention name of the index, the table and column, and whether the index is in ascending or descending order.

Syntax

```
CREATE [UNIQUE] INDEX index_name
ON table_name (column_name1,[column_name2, column_name3...]);
```

For Example

```
1. CREATE INDEX emp_ind1 on Employee(Employee_name);
2. CREATE INDEX emp_ind2 ON Employee(Employee_no, Employee_name);
3. CREATE UNIQUE INDEX emp_ind3 on Employee(Employee_name);
```

B) Displaying Index

To display index information regarding table following query is used.

Syntax

```
Show index from table_name;
```

For Example

```
Show index from Employee;
```

```
mysql> CREATE INDEX emp_ind1 on Employee(Employee_name);
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX emp_ind2 ON Employee(Employee_no, Employee_name);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE UNIQUE INDEX emp_ind3 on Employee(Employee_name);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

```
mysql> show index from Employee;
+----------+------------+----------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name   | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+----------+------------+----------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| employee |          0 | emp_ind3 |            1 | Employee_name | A         |           0 |     NULL | NULL   | YES  | BTREE      |         |               |
| employee |          1 | emp_ind1 |            1 | Employee_name | A         |           0 |     NULL | NULL   | YES  | BTREE      |         |               |
| employee |          1 | emp_ind2 |            1 | Employee_no   | A         |           0 |     NULL | NULL   | YES  | BTREE      |         |               |
| employee |          1 | emp_ind2 |            2 | Employee_name | A         |           0 |     NULL | NULL   | YES  | BTREE      |         |               |
+----------+------------+----------+--------------+---------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
4 rows in set (0.02 sec)

mysql>
```

C) Dropping Index

To drop index of a table following query is used.

Syntax

```
Drop index index_name on table_name;
```

For Example

```
Drop index emp_ind2 on Employee;
```

DDL Commands on Sequence

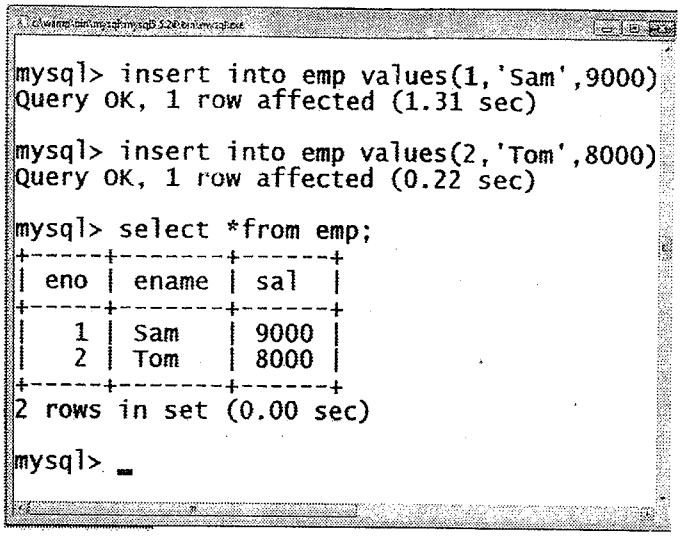
Sequence : A sequence is a set of integers. Sequences are generated in order as per requirement. Sequences are used to create unique values for the rows.

Creating sequence

```
mysql> create table emp(
    -> eno integer(3) auto_increment,
    -> primary key(eno),
    -> ename varchar(20),
    -> sal integer(5));
Query OK, 0 rows affected (0.79 sec)

mysql>
```

Now insert records in the table



```
mysql> insert into emp values(1,'Sam',9000)
Query OK, 1 row affected (1.31 sec)

mysql> insert into emp values(2,'Tom',8000)
Query OK, 1 row affected (0.22 sec)

mysql> select *from emp;
+-----+-------+------+
| eno | ename | sal  |
+-----+-------+------+
|   1 | Sam   | 9000 |
|   2 | Tom   | 8000 |
+-----+-------+------+
2 rows in set (0.00 sec)

mysql> _
```

Here the column eno has auto increment values.

DDL Commands on Synonym

Synonyms are basically alternative names for a table, view, sequence, procedure, stored function, package etc. Synonyms provide both data independence and location transparency.

- Creating Synonym in Oracle
- Create synonym for emp;

Assignment 3 : Design at least 10 SQL queries for suitable database application using SQL DML statements : Insert, Select, Update, Delete with operators, functions, and set operator.

Ans. :

In previous assignment we studied about DDL statements now we will study DML statements.

Data Manipulation Language (DML)

The Data Manipulation Language (DML) is used for accessing and manipulating data in a database. It allows users to access, insert, update, and delete data from the database.

- To insert record into the table – **INSERT**
- To access or read records from table – **SELECT**
- Update the records in table – **UPDATE**
- Delete the records from the table – **DELETE**

We created userdb database previously in that we have a table named as Employee. So no perform DML queries on that table as follows :

1. Inserting record into table

After creation of table the insert command is used to insert one or more records in the table.

Insert query has different forms.

Format 1 : Inserting a single row of data into a table

Syntax

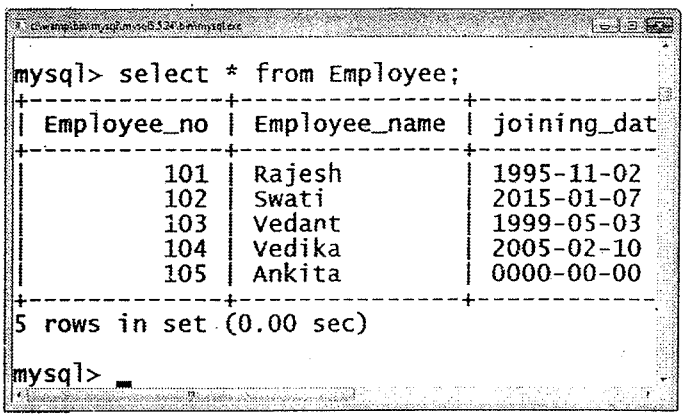
```
Insert into table_name[(column_name1, column_name2,...)]
values (value1, value2....);
```

For example

1. Insert 5 records in Employee table

```
Insert into Employee values
(101,'Rajesh', '1995-11-02',12000),(102,'Swati', '2015-01-07',20000),
(103,'Vedant','1999-05-03',30000), (104,'Vedika','2005-02-10',52000),
(105,'Ankita', '201603-01',8000);
```

After inserting 5 records in table the table will look like as follows :



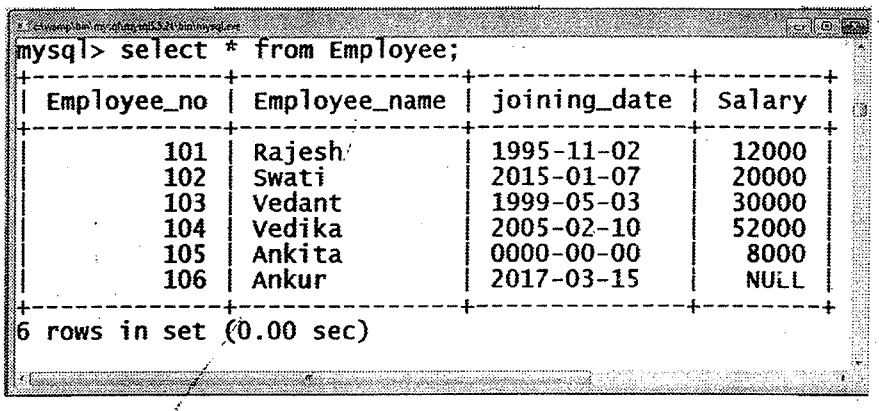
```
mysql> select * from Employee;
+-------------+---------------+------------
| Employee_no | Employee_name | joining_dat
+-------------+---------------+------------
|         101 | Rajesh        | 1995-11-02
|         102 | Swati         | 2015-01-07
|         103 | Vedant        | 1999-05-03
|         104 | Vedika        | 2005-02-10
|         105 | Ankita        | 0000-00-00
+-------------+---------------+------------
5 rows in set (0.00 sec)

mysql>
```

2. Consider we do not have value for salary while inserting a new record in Employee table. Then the query will be

```
Insert into Employee(Employee_no, Employee_name, Joining_date) values(106,'Ankur','2017-03-15');
```

It will set salary value for the employee as NULL.



```
mysql> select * from Employee;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
|         105 | Ankita        | 0000-00-00   |   8000 |
|         106 | Ankur         | 2017-03-15   |   NULL |
+-------------+---------------+--------------+--------+
6 rows in set (0.00 sec)
```

Format 2 : Inserting data into a table from another table

Syntax

```
Insert into table_name select column_name1, columne_name2 from table_name;
```

For Example

- Insert records in newEmployee1 table same as in Employee table.
- Insert into newEmployee1 as select * from Employee;

```
mysql> insert into newEmployee1 select * from Employee;
Query OK, 6 rows affected (0.09 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from newEmployee1;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
|         105 | Ankita        | 0000-00-00   |   8000 |
|         106 | Ankur         | 2017-03-15   |   NULL |
+-------------+---------------+--------------+--------+
6 rows in set (0.00 sec)

mysql>
```

2. Retrieve data from tables

SELECT Query

SELECT query is used to retrieve the data from database. SELECT query never make any change in the database.

Syntax

```
SELECT column_name1, column_name1... from table_name;
```

For Example

Consider the table Employee

```
SELECT Employee_no, Employee_name from Employee;
```

If we Want to retrieve data from all the columns of a table then instead of writing all the column name just use '*'. The '* 'symbol represent all the columns.

For Example

```
SELECT * from Employee;
```

```
mysql> SELECT * from Employee;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
|         105 | Ankita        | 0000-00-00   |   8000 |
|         106 | Ankur         | 2017-03-15   |   NULL |
+-------------+---------------+--------------+--------+
6 rows in set (0.00 sec)

mysql>
```

2.1 With SELECT statement different clauses can be used to display the data as per our requirements

A) WHERE Clause

WHERE clause is used to specify condition in SELECT statement while fetching records from the database. The records satisfying the condition given by where clause are retrieved.

Syntax

```
SELECT column_1,columns_2... from table_name where condition;
```

For Example

```
Select * from Employee where Salary > 8000;
```

```
mysql> select * from Employee where Salary > 8000;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
+-------------+---------------+--------------+--------+
4 rows in set (0.03 sec)
```

```
Select * from Employee where Employee_name='Swati' ;
```

```
mysql> Select * from Employee where Employee_name='Swati';
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         102 | Swati         | 2015-01-07   |  20000 |
+-------------+---------------+--------------+--------+
1 row in set (0.00 sec)

mysql>
```

B) DISTINCT Clause

This clause is used to avoid selection of duplicate rows.

Consider a situation where duplicate values for Salary column are present in Employee table, and the manager wants to know the salary amount given to the employees. To avoid duplication of salary amount we use distinct keyword as follows:

```
Select distinct(Salary) from Employee;
```

```
mysql> Select distinct(Salary) from Employee;
+--------+
| Salary |
+--------+
|  12000 |
|  20000 |
|  30000 |
|  52000 |
|   8000 |
|   NULL |
+--------+
6 rows in set (0.00 sec)

mysql>
```

C) ORDERBY Clause

To arrange the displayed rows in ascending or descending order of given column, ORDERBY Clause is used.

For Example

We need to display the employee information as per their joining dates in ascending order, the query will be

```
mysql> Select * from Employee order by Joining_date;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         105 | Ankita        | 0000-00-00   |   8000 |
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         106 | Ankur         | 2017-03-15   |   NULL |
+-------------+---------------+--------------+--------+
6 rows in set (0.02 sec)

mysql>
```

Now to display same information in descending order on job the query will be

```
Select * from Employee order by Joining_date desc;
```

```
mysql> Select * from Employee order by Joining_date desc;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         106 | Ankur         | 2017-03-15   |   NULL |
|         102 | Swati         | 2015-01-07   |  20000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         105 | Ankita        | 0000-00-00   |   8000 |
+-------------+---------------+--------------+--------+
6 rows in set (0.00 sec)

mysql>
```

3. Modify data in tables

Update

To make changes in the database 'update' command is used. The update command consists of 'set' clause and an optional 'where' clause'. **'WHERE'** clause is used to make changes in specific records.

Syntax

```
Update table_name set column_name = new_value [where condition];
```

For Example

```
Update Employee set Salary=50000 where Employee_no=106;
```

```
mysql> select *from Employee;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
|         105 | Ankita        | 0000-00-00   |   8000 |
|         106 | Ankur         | 2017-03-15   |  50000 |
+-------------+---------------+--------------+--------+
6 rows in set (0.00 sec)

mysql>
```

```
Update Employee set joining_date='2015-03-15' where Employee_no=105;
```

```
mysql> Update Employee set joining_date='2015-03-15' where Employee_no=105;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select *from Employee;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
|         105 | Ankita        | 2015-03-15   |   8000 |
|         106 | Ankur         | 2017-03-15   |  50000 |
+-------------+---------------+--------------+--------+
6 rows in set (0.00 sec)

mysql>
```

4. Remove records from table

Delete

As per requirement, the records from existing table can be removed using delete command. Delete command can have 'WHERE' clause optionally.

Syntax

```
Delete from table_name;
```

For Example

- Write a query to remove record of Employee_no 106 from Employee table,

```
Delete from Employee where Employee_no = 106;
```

```
mysql> Delete from Employee where Employee_no = 106;
Query OK, 1 row affected (0.06 sec)

mysql> select *from Employee;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
|         105 | Ankita        | 2015-03-15   |   8000 |
+-------------+---------------+--------------+--------+
5 rows in set (0.00 sec)

mysql>
```

2.2 Between, In, Like operators are also used with SELECT statement to display the data as per our requirements

A) Between Predicate

Between predicate is used to specify certain range of values. The AND keyword is used in this predicate.

Syntax

```
test_expression [ NOT ] BETWEEN begin_expression AND end_expression
```

For Example

```
Select * from Employee where Salary between 10000 and 20000;
```

```
mysql> Select * from Employee where Salary between 10000 and 20000;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
+-------------+---------------+--------------+--------+
2 rows in set (0.01 sec)

mysql>
```

```
select * from Employee where Salary not between 10000 and 20000;
```

```
mysql> Select * from Employee where Salary between 10000 and 20000;
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
+-------------+---------------+--------------+--------+
2 rows in set (0.01 sec)

mysql>
```

B) In Predicate

IN predicate particularly determines whether the value of expression given to test matches any value in specified the list.

For Example

- Manager wants to view records of employees whose salary is 8000 and 30000.
- Then the query will be

```
Select * from Employee where Salary in(8000,30000);
```

```
mysql> select * from Employee where Salary in(8000,30000);
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         103 | Vedant        | 1999-05-03   |  30000 |
|         105 | Ankita        | 2015-03-15   |   8000 |
+-------------+---------------+--------------+--------+
2 rows in set (0.00 sec)

mysql>
```

- Manager wants to view records of employees whose salary is not 8000 and 30000.
- Then the query will be

```
Select * from Employee where Salary not in(8000,30000);
```

```
mysql> Select * from Employee where Salary not in(8000,30000);
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         102 | Swati         | 2015-01-07   |  20000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
+-------------+---------------+--------------+--------+
3 rows in set (0.14 sec)
```

C) Like Predicate

Like operator determines whether a specific character string matches the given pattern or not.

For Example

- Display records of employee who join in 2015

```
select * from Employee where joining_date like '2015%';
```

```
mysql> select * from Employee where joining_date like '2015%';
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         102 | Swati         | 2015-01-07   |  20000 |
|         105 | Ankita        | 2015-03-15   |   8000 |
+-------------+---------------+--------------+--------+
2 rows in set, 1 warning (0.00 sec)

mysql>
```

Display records of employee who didn't join in 2015

```
select * from Employee where joining_date not like '2015%';
```

```
mysql> select * from Employee where joining_date not like '2015%';
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         101 | Rajesh        | 1995-11-02   |  12000 |
|         103 | Vedant        | 1999-05-03   |  30000 |
|         104 | Vedika        | 2005-02-10   |  52000 |
+-------------+---------------+--------------+--------+
3 rows in set, 1 warning (0.00 sec)

mysql>
```

Aggregate Functions used in SQL Queries to perform DML operations on table data

i) Min()

This function returns smallest value from specified column of the table.

For Example

- Write a query to retrieve record of employee who gets least salary as compare to other employees.

```
Select * from Employee where Salary=(Select min(Salary) from Employee);
```

```
+-------------+---------------+--------------+--------+
3 rows in set, 1 warning (0.00 sec)

mysql> Select * from Employee where Salary=(Select min(Salary) from Employee);
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         105 | Ankita        | 2015-03-15   |   8000 |
+-------------+---------------+--------------+--------+
1 row in set (0.03 sec)

mysql>
```

ii) Max()

This function returns greatest value from specified column of the table.

For Example

– Write a query to retrieve record of employee who gets maximum salary as compare to other employees.

```
Select * from Employee where Salary=(Select max(Salary) from Employee);
```

```
+-------------+---------------+--------------+--------+
1 row in set (0.03 sec)

mysql> Select * from Employee where Salary=(Select max(Salary) from Employee);
+-------------+---------------+--------------+--------+
| Employee_no | Employee_name | joining_date | Salary |
+-------------+---------------+--------------+--------+
|         104 | Vedika        | 2005-02-10   |  52000 |
+-------------+---------------+--------------+--------+
1 row in set (0.00 sec)

mysql>
```

iii) Count()

This function returns total number of values of specified column of the table.

For Example

– Write a query to retrieve count of employees who join in 2015 year.

```
select count(Employee_no) from Employee where joining_date like '2015%';
```

```
mysql> select count(Employee_no)  from Employee where joining_date  like '2015%'
;
+--------------------+
| count(Employee_no) |
+--------------------+
|                  2 |
+--------------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

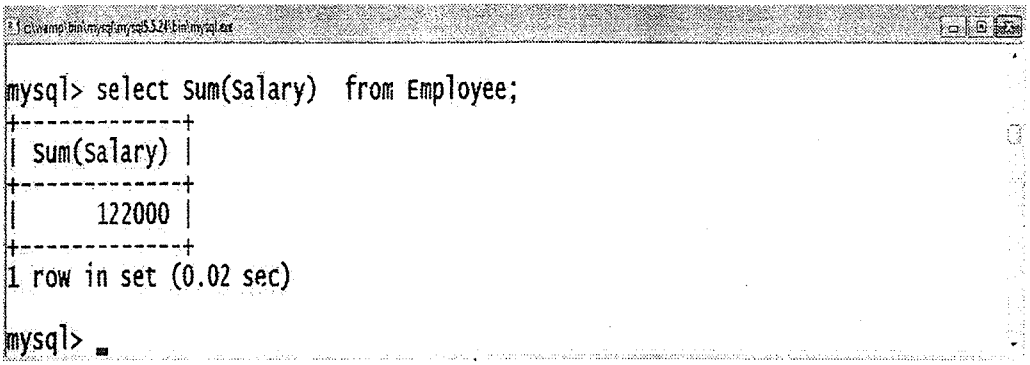
iv) Sum()

This function returns sum of all the values of specified column of the table.

For Example

– Write a query to to find total salary amount paid to all the employees.

```
select Sum(Salary)  from Employee;
```



```
mysql> select Sum(Salary)  from Employee;
+-------------+
| Sum(Salary) |
+-------------+
|      122000 |
+-------------+
1 row in set (0.02 sec)

mysql>
```

Scalar Functions used in SQL Queries to perform DML operations on table data.

i) MID()

MID function is used to extract substrings from column values of string type in a table.

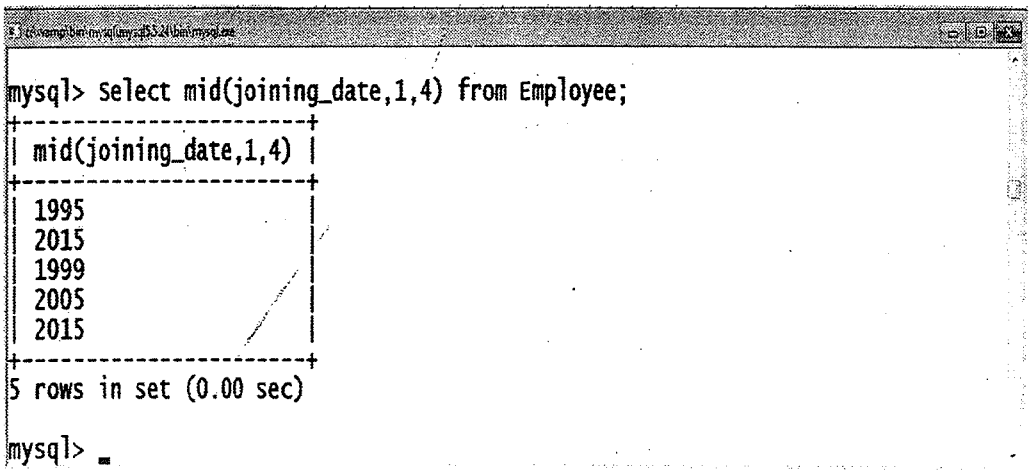
Syntax

```
Select MID(column_name, start, length) from table_name;
```

For Example

– Write a query to dispay only joining year of the employees

```
Select mid(joining_date,1,4) from Employee;
```



```
mysql> Select mid(joining_date,1,4) from Employee;
+----------------------+
| mid(joining_date,1,4) |
+----------------------+
| 1995                 |
| 2015                 |
| 1999                 |
| 2005                 |
| 2015                 |
+----------------------+
5 rows in set (0.00 sec)

mysql>
```

ii) LCASE()

LCASE function is used to convert value of string column to Lowecase character.

Syntax

```
Select LCASE(column_name) from table_name;
```

For Example

– Write a query to convert employee name to lowercase.

```
Select lcase(Employee_name) from Employee;
```

```
mysql> Select lcase(Employee_name) from Employee;
+----------------------+
| lcase(Employee_name) |
+----------------------+
| rajesh               |
| swati                |
| vedant               |
| vedika               |
| ankita               |
+----------------------+
5 rows in set (0.00 sec)

mysql>
```

iii) UCASE()

UCASE function is used to convert value of string column to upper case character.

Syntax

```
Select UCASE(column_name) from table_name;
```

For Example

- Write a query to convert employee name to lowercase.

```
Select ucase(Employee_name) from Employee;
```

```
mysql> Select ucase(Employee_name) from Employee;
+----------------------+
| ucase(Employee_name) |
+----------------------+
| RAJESH               |
| SWATI                |
| VEDANT               |
| VEDIKA               |
| ANKITA               |
+----------------------+
5 rows in set (0.00 sec)

mysql>
```

iv) ROUND()

ROUND function is used to round a numeric field to number of nearest integer. It is used on Decimal point values.

Syntax

```
Select round(column_name) from table_name;
```

For Example

- Suppose employee table has salary in decimal point to round it we can use round function as:

```
Select round(Salary) from Employee;
```

Set operations in SQL Queries to perform DML operations on table data.

- The different Set Operators are as follows
  - Union
  - Union All
  - Intersect
  - Minus

Consider following two tables product_details and sales_details

```
mysql> select * from product_details;
+------------+--------------+----------+-------+
| Product_id | Product_name | Quantity | price |
+------------+--------------+----------+-------+
|       5001 | Pendrive     |      100 |   900 |
|       5002 | Harddisk     |      200 |  3500 |
|       5003 | Headphone    |     1000 |   600 |
|       5004 | DVD          |       20 |  1500 |
|       5005 | speaker      |      600 |  1200 |
|       5006 | headphone    |     1000 |   400 |
+------------+--------------+----------+-------+
6 rows in set (0.00 sec)
```

```
mysql> select * from sales_details;
+---------+------------+----------+-------+---------------+
| Sale_no | product_id | Quantity | price | customer_name |
+---------+------------+----------+-------+---------------+
|    2001 |       5001 |       50 |   900 | Savni         |
|    2002 |       5004 |       10 |  1500 | Savni         |
|    2003 |       5003 |      120 |   600 | Deepak        |
|    2004 |       5005 |      420 |  1200 | Harsh         |
|    2005 |       5002 |       40 |  3500 | Akash         |
+---------+------------+----------+-------+---------------+
5 rows in set (0.00 sec)

mysql>
```

i) Union

The union operator returns all distinct rows selected by either query

Syntax

```
Select column_name from table_1
Union
Select column_name from table_2
```

For Example

– Write a query to retrieve ids of all the products even if they were sale or present in the storage room.

```
Select Product_id from product_details
union
select Product_id from sales_details;
```

```
mysql> Select Product_id from product_details
    -> union
    -> select Product_id from sales_details;
+------------+
| Product_id |
+------------+
|       5001 |
|       5002 |
|       5003 |
|       5004 |
|       5005 |
|       5006 |
+------------+
6 rows in set (0.00 sec)

mysql>
```

ii) Union All

The Union All operator returns all rows selected by either query including duplicates.

Syntax

```
Select column_name from table_1
Union all
Select column_name from table_2
```

For Example

– Write a query to retrieve ids of all the products even if they were sold or present in the storage room.

```
Select Product_id from product_details
Union all
select Product_id from sales_details;
```

```
mysql> Select Product_id from product_details
    -> Union all
    -> select Product_id from sales_details;
+------------+
| Product_id |
+------------+
|       5001 |
|       5002 |
|       5003 |
|       5004 |
|       5005 |
|       5006 |
|       5001 |
|       5004 |
|       5003 |
|       5005 |
|       5002 |
+------------+
11 rows in set (0.00 sec)
```

iii) Intersect

The intersect operator returns only those rows which are common to both the queries

Syntax

```
Select column_name from table_1
intersect
Select column_name from table_2
```

For Example

– Write a query to retrieve ids of all the sold products.

```
Select Product_id from product_details
intersect
select Product_id from sale_details;
```

iv) Minus

Minus operator displays the rows which are present in the first query but absent in the second query, with no duplicates and data is arranged in ascending order by default.

Syntax

```
Select column_name from table_1
minus
Select column_name from table_2
```

For Example

```
Select Product_id from product_details
minus
select Product_id from sales_details;
```

Assignment 4 : **Design at least 10 SQL queries for suitable database application using SQL DML statements : all types of Join, Sub-Query and View.**

Ans. :

JOINS

To understand joins consider following two tables.

```
mysql> select * from product_details;
+------------+--------------+----------+-------+
| Product_id | Product_name | Quantity | price |
+------------+--------------+----------+-------+
|       5001 | Pendrive     |      100 |   900 |
|       5002 | Harddisk     |      200 |  3500 |
|       5003 | Headphone    |     1000 |   600 |
|       5004 | DVD          |       20 |  1500 |
|       5005 | speaker      |      600 |  1200 |
|       5006 | headphone    |     1000 |   400 |
+------------+--------------+----------+-------+
6 rows in set (0.00 sec)
```

```
mysql> select * from sales_details;
+---------+------------+----------+-------+---------------+
| Sale_no | product_id | Quantity | price | customer_name |
+---------+------------+----------+-------+---------------+
|    2001 |       5001 |       50 |   900 | Savni         |
|    2002 |       5004 |       10 |  1500 | Savni         |
|    2003 |       5003 |      120 |   600 | Deepak        |
|    2004 |       5005 |      420 |  1200 | Harsh         |
|    2005 |       5002 |       40 |  3500 | Akash         |
+---------+------------+----------+-------+---------------+
5 rows in set (0.00 sec)

mysql>
```

1) Inner Join (Equi Join)

The INNER JOIN is used to display records that have matching values in both tables.

Syntax :

```
Select column_name_list from table_1
INNER JOIN table_2
where table_1.column_name = table_2.column_name
```

For Example :

```
select product_details.Product_id, Product_name, customer_name, sale_details.Quantity,
product_details.price from product_details INNER JOIN sales_details on
product_details.Product_id=sales_details.Product_id ;
```

```
mysql> select  product_details.Product_id, Product_name, customer_name, sale_det
ails.Quantity, product_details.price from product_details INNER JOIN sale_detail
s on product_details.Product_id=sale_details.Product_id ;
ERROR 1146 (42S02): Table 'userdb.sale_details' doesn't exist
mysql> select  product_details.Product_id, Product_name, customer_name, sales_de
tails.Quantity, product_details.price from product_details INNER JOIN sales_deta
ils on product_details.Product_id=sales_details.Product_id ;
+------------+--------------+---------------+----------+-------+
| Product_id | Product_name | customer_name | Quantity | price |
+------------+--------------+---------------+----------+-------+
|       5001 | Pendrive     | Savni         |       50 |   900 |
|       5002 | Harddisk     | Akash         |       40 |  3500 |
|       5003 | Headphone    | Deepak        |      120 |   600 |
|       5004 | DVD          | Savni         |       10 |  1500 |
|       5005 | speaker      | Harsh         |      420 |  1200 |
+------------+--------------+---------------+----------+-------+
5 rows in set (0.00 sec)
```

2) Outer Join

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

(i) Left Outer Join (ii) Right Outer Join (iii) Full Outer Join

(i) Left Outer Join

The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. Null values are shown at the place of right table values.

Syntax

```
SELECT column-name-list
from table-name1
LEFT OUTER JOIN
table-name2
on table-name1.column-name = table-name2.column-name;
```

For Example

```
SELECT product_details.Product_id, product_details.Product_name, sales_details.customer_name FROM
product_details LEFT OUTER JOIN sales_details ON sales_details.Product_id =product_details.Product_id;
```

```
mysql>  SELECT product_details.Product_id, product_details.Product_name, sales_d
etails.customer_name FROM product_details LEFT OUTER JOIN sales_details ON sales
_details.Product_id =product_details.Product_id;
+------------+--------------+---------------+
| Product_id | Product_name | customer_name |
+------------+--------------+---------------+
|       5001 | Pendrive     | Savni         |
|       5002 | Harddisk     | Akash         |
|       5003 | Headphone    | Deepak        |
|       5004 | DVD          | Savni         |
|       5005 | speaker      | Harsh         |
|       5006 | headphone    | NULL          |
+------------+--------------+---------------+
6 rows in set (0.00 sec)
```

(ii) Right Outer Join

Returns all rows from the right table even if there are no matches in the left table. Null values are shown at the place of left table values.

Syntax

```
select column-name-list
from table-name1
RIGHT OUTER JOIN
table-name2
on table-name1.column-name = table-name2.column-name;
```

For Example

```
SELECT product_details.Product_id, product_details.Product_name, sales_details.customer_name FROM
product_details RIGHT OUTER JOIN sales_details ON sales_details.Product_id =product_details.Product_id;
```

```
mysql> SELECT product_details.Product_id, product_details.Product_name, sales_de
tails.customer_name FROM product_details RIGHT OUTER JOIN sales_details ON sales
_details.Product_id =product_details.Product_id;
+------------+--------------+---------------+
| Product_id | Product_name | customer_name |
+------------+--------------+---------------+
|       5001 | Pendrive     | Savni         |
|       5004 | DVD          | Savni         |
|       5003 | Headphone    | Deepak        |
|       5005 | speaker      | Harsh         |
|       5002 | Harddisk     | Akash         |
+------------+--------------+---------------+
5 rows in set (0.02 sec)

mysql>
```

(iii) Full Outer Join

The full outer join returns a result with the **matching data** of both the tables and then remaining rows of both **left** table and then the **right** table.

Sub-Queries

Writing a query inside another query is known as nested query or subquery. The inner query get executed first, then the output on inner query is given as input to outer query.

Consider the previous emp table

Example

To display records of employees working in SMITH's department

```
Select * from emp where deptno =
(select deptno from emp where ename = 'SMITH');
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7876	ADAMS	CLERK	7788	01/12/1983	1100		20

(1) List all the sold product names;

```
Select product_details.Product_name from product_details where product_details.Product_id IN (select
sales_details.Product_id from sales_details);
```

```
mysql> Select product_details.Product_name from product_details where product_de
tails.Product_id IN (select sales_details.Product_id from sales_details);
+--------------+
| Product_name |
+--------------+
| Pendrive     |
| Harddisk     |
| Headphone    |
| DVD          |
| speaker      |
+--------------+
5 rows in set (0.00 sec)

mysql> _
```

Note : Assignment solved with Oracle.

Assignment 5 : **Unnamed PL/SQL code block: use of control structure and Exceptions handling is mandatory. Write a PL/SQL block of code for the following requirements :-**
Schema
(1) Borrower(Rollin, Name, DateofIssue, NameofBook, Status)
(2) Fine(Roll_no, Date, Amt)
Accept Roll_no and name of book from user.
Check the number of days (from date of issue), if days are between 15 and 30then fine amount will be Rs.5 per day.
If no. of days>30, per day fine will be Rs.50 per day and for days less than 30, Rs. 5 per day.
After submitting the book, status will be change from I to R.
If condition of fine is true, then details will be stored into fine table.

Ans. :

Create borrower table

```
create table borrower(roll_no number(5), name varchar2(20),DateofIssue date,NameofBook
varchar2(20),status varchar2(20));
```

Insert records in borrower table

```
insert into borrower values(1,'Rahul','01-Apr-2017','DBMS','I');;
insert into borrower values(2,'Kunal','05-Apr-2017','Java','I');
insert into borrower values(3,'Sarika','10-Apr-2017','OS','I');
```

```
SQL> select * from borrower;

   ROLL_NO NAME                 DATEOFISS NAMEOFBOOK          STATUS
---------- -------------------- --------- -------------------- --------------------
         1 Rahul                01-APR-17 DBMS                 I
         2 Kunal                05-APR-17 Java                 I
         3 Sarika               10-APR-17 OS                   I
```

Create Fine table

```
create table fine(roll_no number(5),sdate date,Amt number(5));
```

PL-SQL Block

```
Declare
rno number(3) := &roll_number;
bname varchar2(20) := '&book_name';
no_days number(7);
issuedate date;
fineamt number(5):=0;
begin
select DateofIssue into issuedate from borrower where roll_no = rno;
select sysdate - to_date(issuedate) days into no_days from dual;
if (no_days >=15 and no_days <=30) then
fineamt := no_days * 5;
elsif no_days>30 then
fineamt := no_days * 50;
end if;
update borrower set status = 'R' where roll_no = rno;
if fineamt >= 0 then
```

```
insert into fine values(rno,sysdate,fineamt);
end if;
EXCEPTION
WHEN no_data_found THEN
dbms_output.put_line('Record not found');
end;
/
```

Run The PLSQL Block

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> /
Enter value for roll_number: 1
old   2: rno number(3) := &roll_number;
new   2: rno number(3) := 1;
Enter value for book_name: DBMS
old   3: bname varchar2(20) := '&book_name';
new   3: bname varchar2(20) := 'DBMS';

PL/SQL procedure successfully completed.
```

Verify Both tables

```
SQL> select * from borrower;

  ROLL_NO NAME                 DATEOFISS NAMEOFBOOK           STATUS
--------- -------------------- --------- -------------------- --------------------
        1 Rahul                01-APR-17 DBMS                 R
        2 Kunal                05-APR-17 Java                 I
        3 Sarika               10-APR-17 OS                   I

SQL> select * from fine;

  ROLL_NO SDATE          AMT
--------- --------- ---------
        1 27-APR-17       135
```

Assignment 6 : **Cursors : (All types: Implicit ,Explicit, Cursor FOR Loop, Parameterized cursor)**

Write PL/SQL block of code using parameterized cursor, that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

Ans. :

For theory refer Section 2.19.1

Create two tables

O_RollCall

```
create table O_RollCall(roll_no number(3), name varchar2(20))
insert records
```

```
  1* select * from O_RollCall
SQL> /.

  ROLL_NO NAME
--------- --------------------
        1 Rahul
        2 Kunal
        2 Sarika
```

N_RollCall

```
create table N_RollCall(roll_no number(3), name varchar2(20))
```

```
  1* select * from N_Rollcall
SQL> /

  ROLL_NO NAME
--------- --------------------
        1 Rahul
```

PL-SQL Block

```
declare
cursor cur1 is
select roll_no,name from O_Rollcall;

cursor cur2 is
select roll_no from N_Rollcall;


r number(3);
rno number(3);
nm varchar2(20);

begin
open cur1;
open cur2;
loop
fetch cur1 into rno,nm;
fetch cur2 into r;

exit when cur1%found = false;
if r <> rno then
insert into N_Rollcall values(rno,nm);
end if;

end loop;
close cur1;
end;
/
```

Assignment 7 : PL/SQL Stored Procedure and Stored Function.

Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and 900 category is first class, if marks 899 and 825 category is Higher second Class.

Write PL/SQL block for using procedure created with above requirement.

Stud_Marks(name, total_marks) Result(Roll, Name, Class)

Ans. :

For Theory refer Section 2.18

Create table stud_marks

```
create table stud_marks(name varchar2(20),total_marks number(5));
create table result(roll number(3), name varchar2(20),class varchar2(20))
```

PL-SQL Block

```
create or replace procedure proc_Grade(rno number,name varchar2,marks number)
is
class varchar2(20);

begin
if (marks <=1500 and marks >=990) then
class := 'Distinction';
elsif(marks <=989 and marks >=900) then
class := 'First';
elsif(marks <=899 and marks >=825) then
class := 'Higher second';
end if;

insert into stud_marks values(name,marks);
insert into result values(rno,name,class);

end;

/
```

Call the procedure

```
SQL> exec proc_Grade(1,'Sam',1000);

PL/SQL procedure successfully completed.

SQL> select * from stud_marks;

NAME                 TOTAL_MARKS
-------------------- -----------
Sam                         1000

SQL> select * from result;

     ROLL NAME                 CLASS
--------- -------------------- --------------------
        1 Sam                  Distinction
```

Assignment 8 : **Database trigger (All types : Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The system should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit Table.**

Ans. :

For Theory refer Section 2.19.2

Create table library

```
create table library(bno number(5), bname varchar2(20),author varchar2(20),allowed_days number(5))
```

Insert some records

```
SQL> select * from library;

       BNO BNAME                AUTHOR               ALLOWED_DAYS
---------- -------------------- -------------------- ------------
         1 Java                 Mr. Patil                      10
         2 VB                   Mr. Sharma                     15
         3 CPP                  Mr. Surve                      15
```

Create table library_audit

```
create table library_audit(bno number(5),old_all_days number(5),new_all_days number(5))
```

```
create or replace trigger tr1
before update or delete on library
for each row

begin
insert into library_audit values(:new.bno,:old.allowed_days,:new.allowed_days);
end;
/
```

Checking of Update Query

```
SQL> select * from library;

       BNO BNAME                AUTHOR               ALLOWED_DAYS
---------- -------------------- -------------------- ------------
         1 Java                 Mr. Patil                      10
         2 VB                   Mr. Sharma                     15
         3 CPP                  Mr. Surve                      15

SQL> update library set allowed_days = 15 where bno = 1;

1 row updated.

SQL> select * from library;

       BNO BNAME                AUTHOR               ALLOWED_DAYS
---------- -------------------- -------------------- ------------
         1 Java                 Mr. Patil                      15
         2 VB                   Mr. Sharma                     15
         3 CPP                  Mr. Surve                      15

SQL> select * from library_audit;

       BNO OLD_ALL_DAYS NEW_ALL_DAYS
---------- ------------ ------------
         1           10           15
```

Checking of Delete Query

```
SQL> delete from library where bno = 1;

1 row deleted.

SQL> select * from library;

      BNO BNAME                AUTHOR               ALLOWED_DAYS
--------- -------------------- -------------------- ------------
        2 VB                   Mr. Sharma                     15
        3 CPP                  Mr. Surve                      15

SQL> select * from library_audit;

      BNO OLD_ALL_DAYS NEW_ALL_DAYS
--------- ------------ ------------
        1           10           15
                    15
```

Group B : Large Scale Database

Assignment 9 : Study of Open Source NOSQL Database : MongoDB(Installation, basic CRUD operations, Execution)

Ans. :

MongoDB installation

Let us see the steps :

- **Step 1 : Download mongodb**
- **Step 2 : Install and configure MOngoDB**
- **Step 3 : Install mongodb as service**

Let us see the steps in details :

Step 1 : Download mongodb

Download Mongodb for windows or Linux (32 bit / 64 bit) from following location

http://www.mongodb.org/downloads

1.1 In this guide we are going to install mongodb 2.4.9 for windows 7 32 bit, download mongodb for windows from below download location

http://fastd1.mongodb.org /win32/mongodb-win32-i386-2.4.9.zip

1.2 Move mongodb-win32-i386-2.4.9.zip file from default download location to c :\

1.3 Extract the compressed source file on the same location i.e. on c:\

1.4 Rename the long folder name to shorter one mongodb-win32-i386-2.4.9 to mongodb

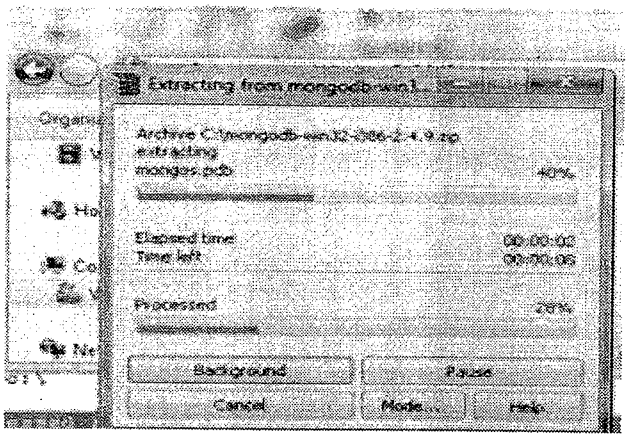
1.5 Create the data directory under c:\mongodb\data\db

1.6 From the command prompt execute below command

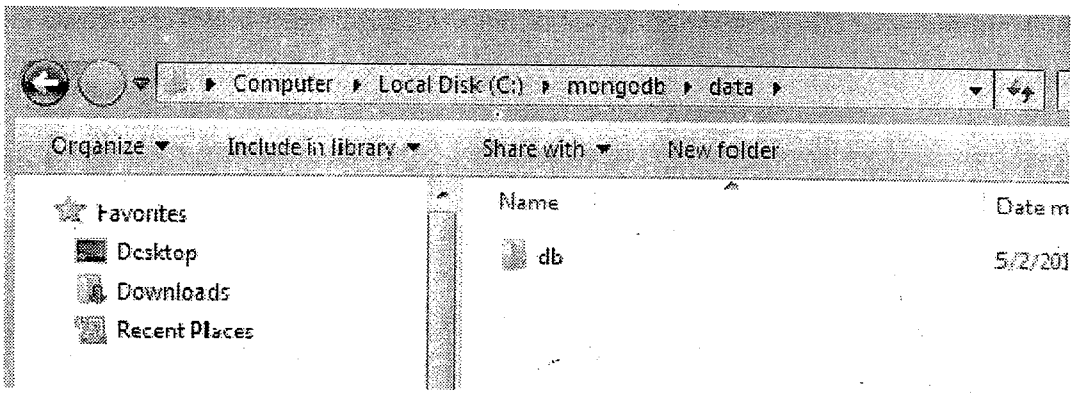
c:\mongodb\bin\mongod.exe-dbpath c:\mongodb\data\db

Note : dbpath option to specify an alternate path for database files

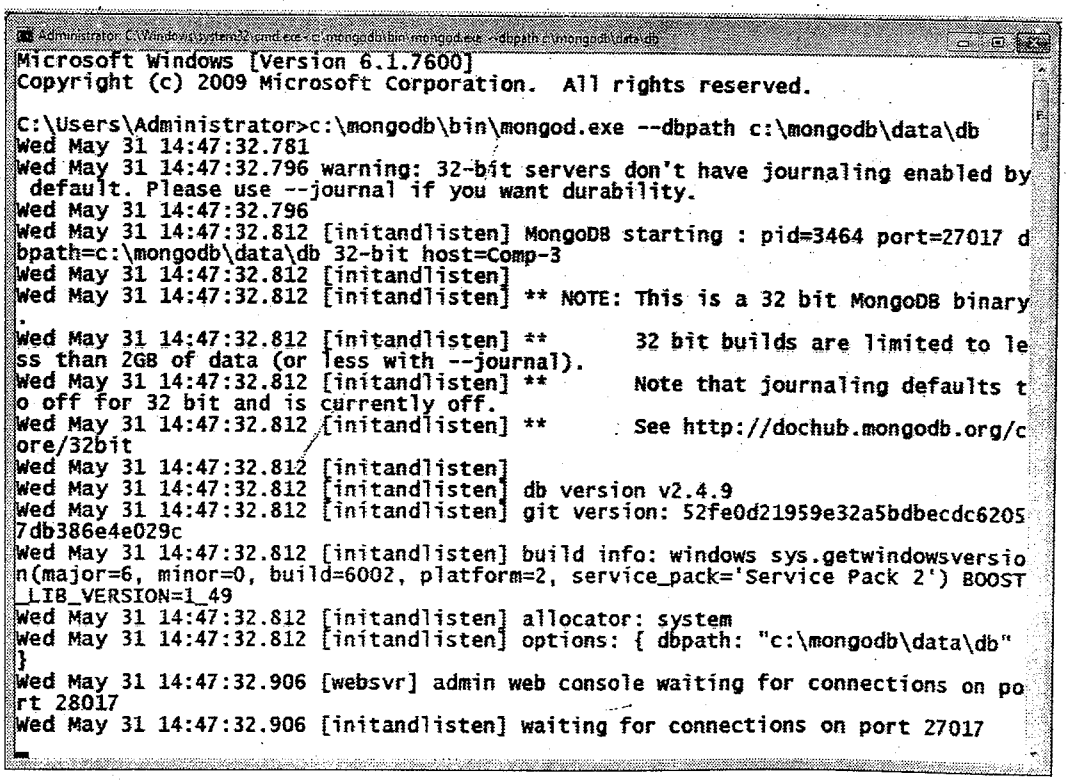
1.3



1.5



1.6



Step 2 : Install and configure MongoDB

Invoke another cmd command window to establish connection to database terminal shell

c:\mongodb\bin\mongo

>show dbs;

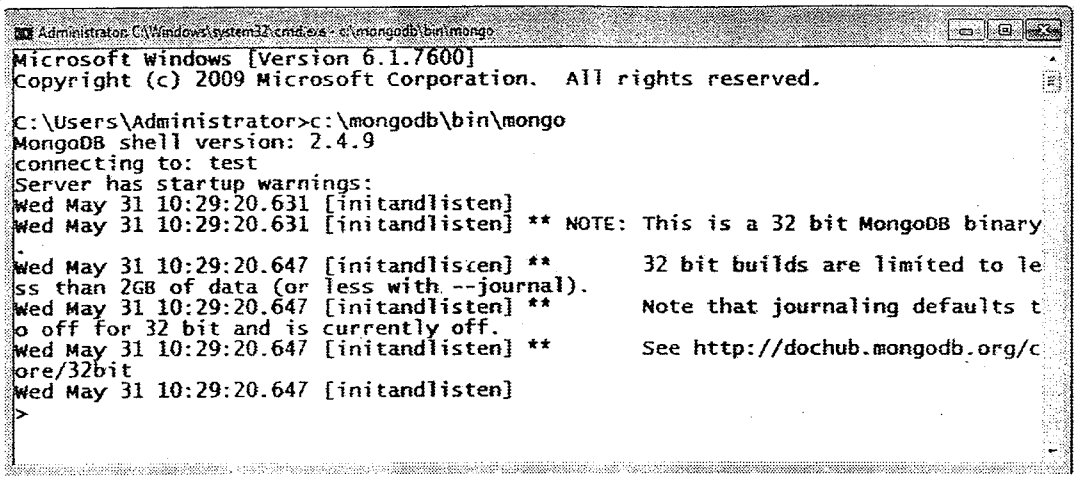
2.1 Set the context to exported by using-below command use expertdb

2.2 Insert the site and owner name doc's in a testingdata collection using below commands

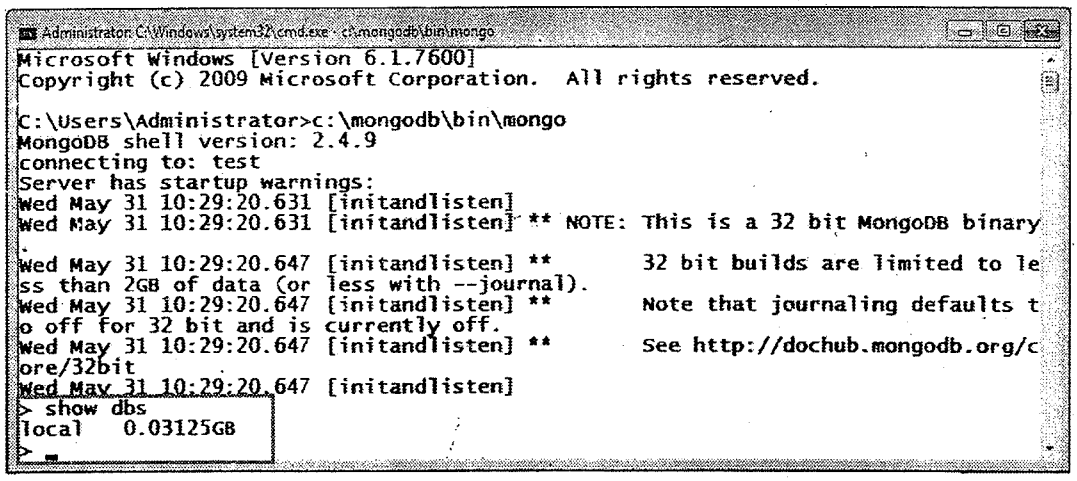
a = (site : "solutionsatexperts.com")

b = (owner : "TMS")

2.1

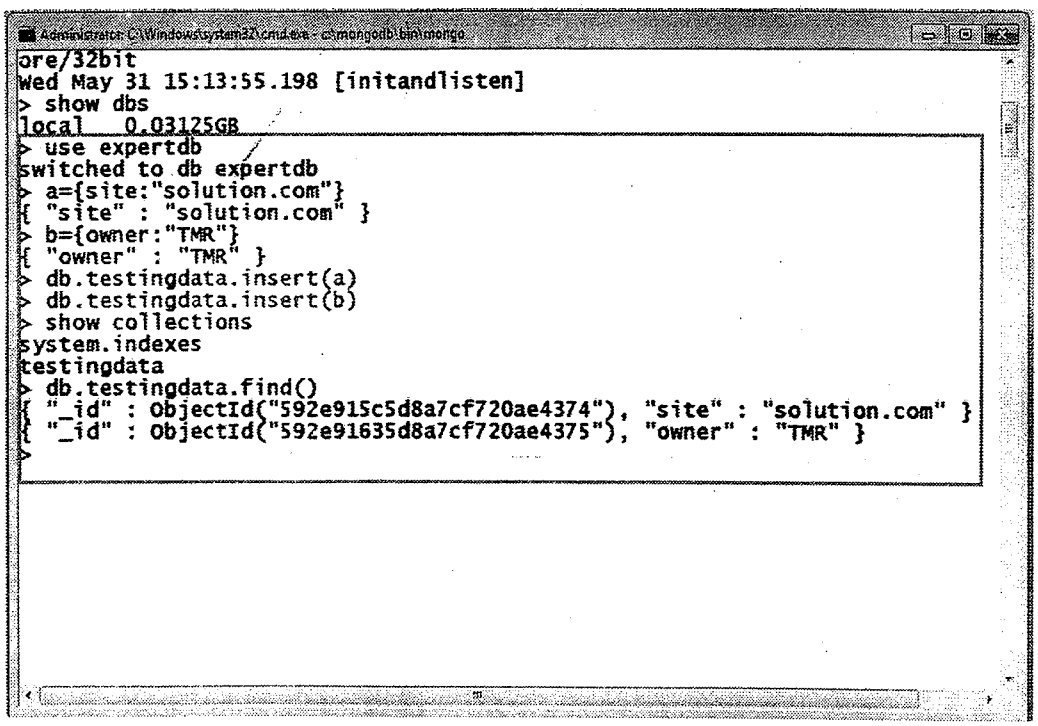


2.2



show collections

Verify the docs exost om testingdata by executing below command db.testingdata.find()



Step 3 : Install and Run the MongoDB as Service

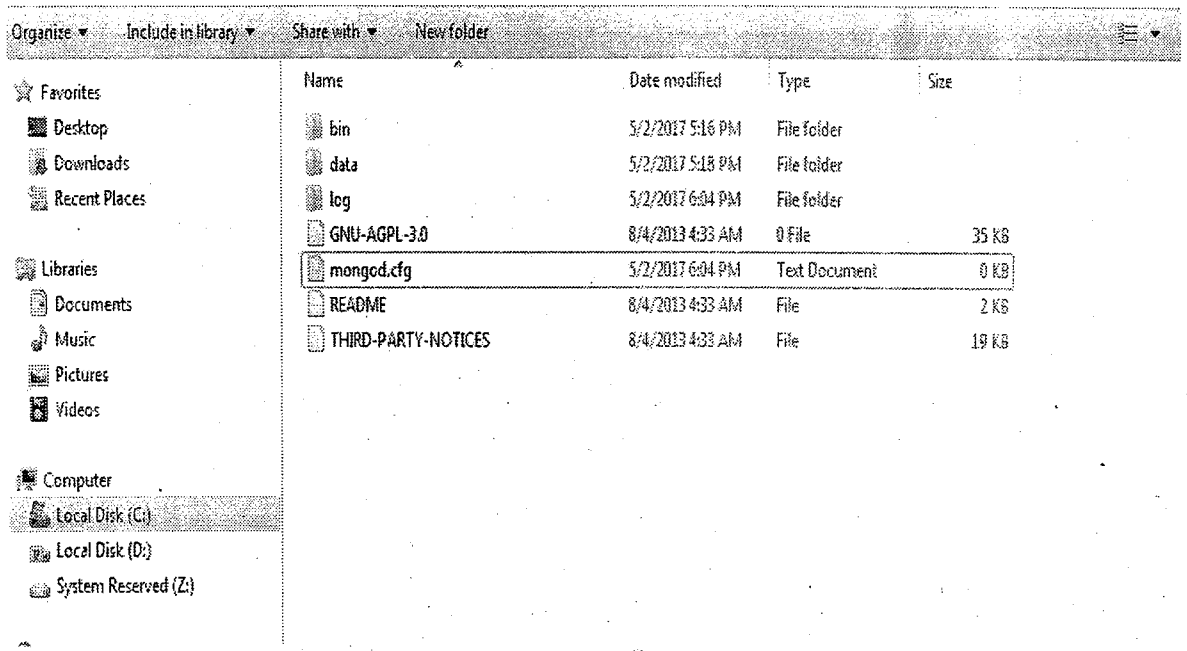
3.1 Open cmd terminal as administrator privilege and create folder "log" under c:\mongodb

c:\mongodb>md log

3.2 Create mongod configuration file under c:\mongodb and add below lines

edit mongod.cfg

3.1



logpath=c:\mongodb\log\mongo.log

dbpath=c:\mongodb\data\db

bind_ip=192.168.0.101

port=27017

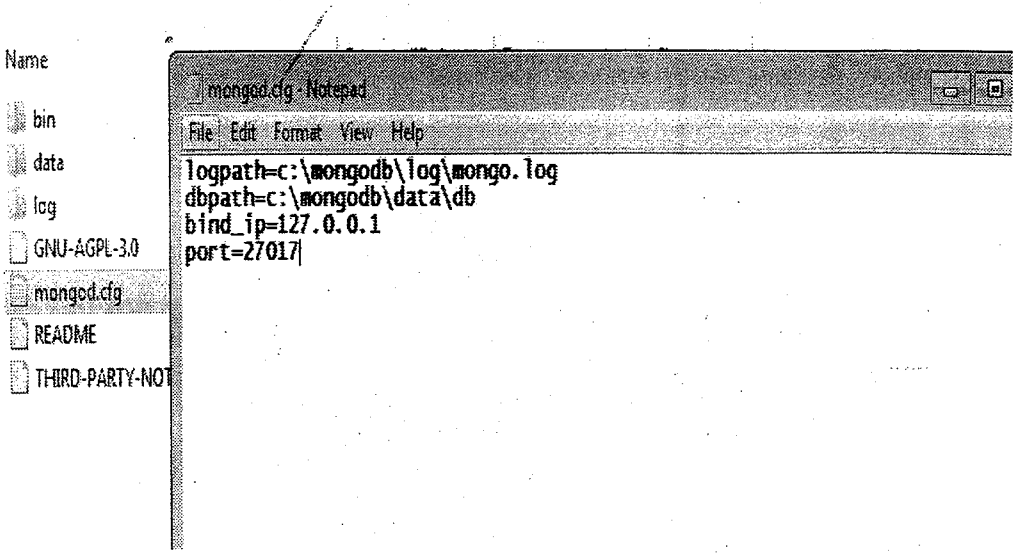
save & Exit.

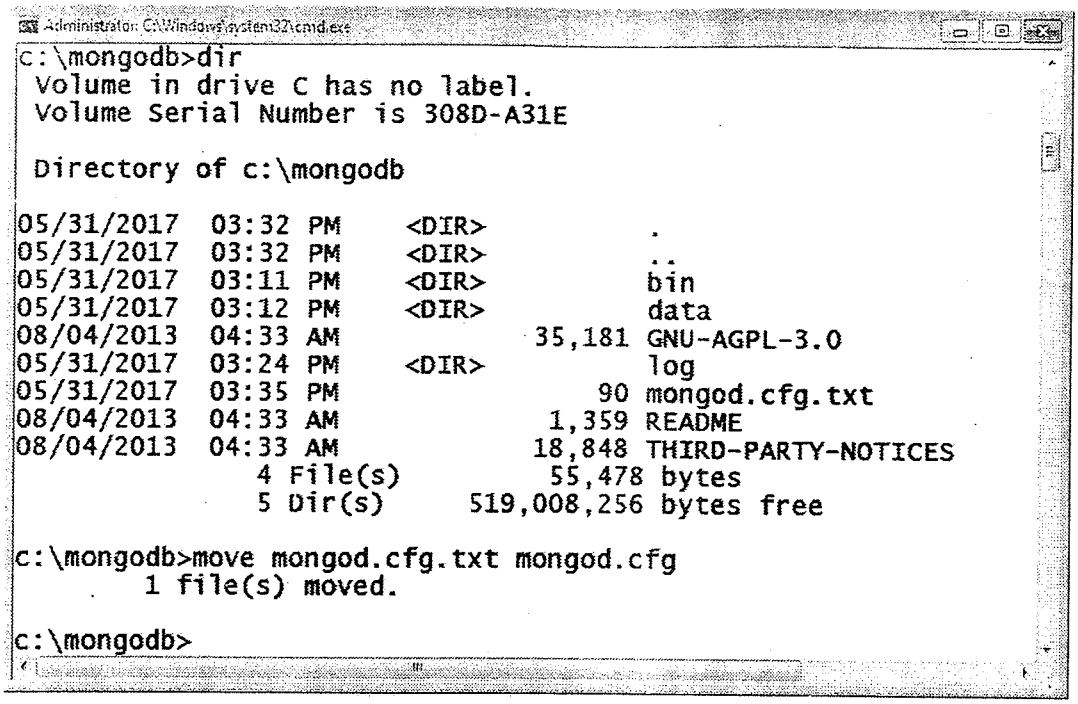
3.3 Install mongodb as service

c:\mongodb\bin\mongod.exe-config c:\mongodb\mongod.cfg-install

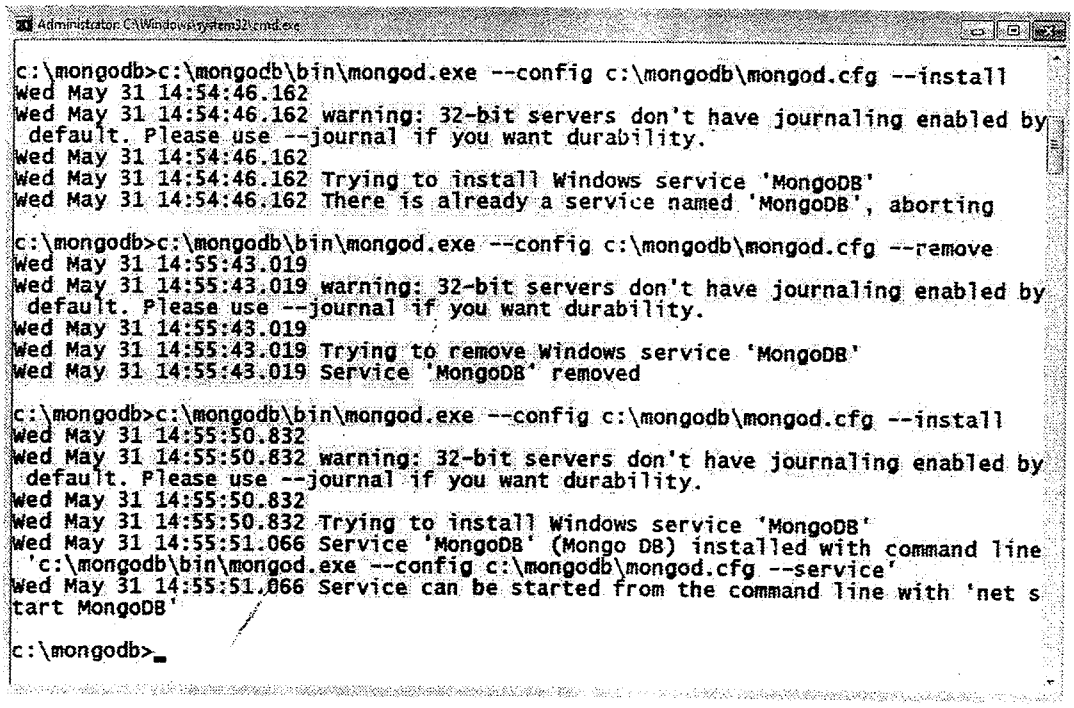
3.4 Start/stop mongodb service using net command line utility

3.2





3.3



net start MongodB

To stop mongodb service use command :

net stop MongoDB

3.5 Invoke another cmd command window to establish connection to database terminal shell

c:\mongodb\bin\mongo

show dbs;

To remove mongodb service use below command

C:\mongodb\bin\mongod.exe - remove

❑❑❑

Note

University In-Sem Exam

Questions and Answers

Questions and Answers for In-Semester Examination

Chapter 1 : Introduction to DBMS

Q. 1.1 A university registrar's office maintains data about the following entities:

- Courses, including number, title, credits, syllable, and prerequisites;
- Courses, offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
- Students including student_id, name, and program; and
- Instructors, including identification number, name, department and title;

Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

(i) Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.

(ii) Construct appropriate tables for E-R diagram designed with above requirements.

(Aug. 2017, 10 marks)

Ans. : (i)

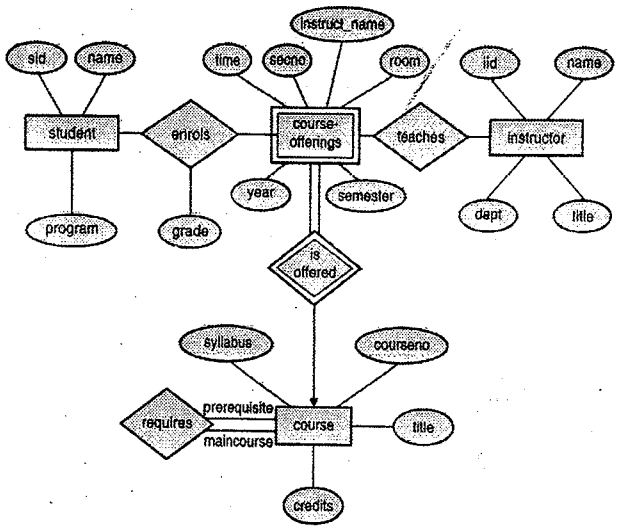


Fig. 1-Q. 1.1(i) : E-R diagram for university

(ii)

- Student(sid, name, program)
- Course (Courseno, title, credits, syllable, prerequisite)
- Instructor(iid, name, dept ,title)
- Course offerings (courseno, year, semester, secno, time, room)

Q. 1.2 Explain various Data Models used in DBMS.

(10 Marks)

Ans. :

There are different types of data models :

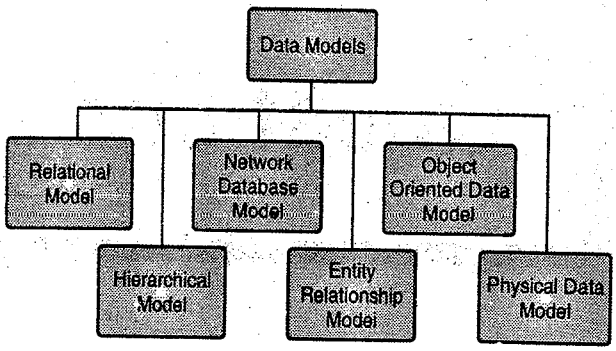


Fig. 1–Q. 1.2

1. Relational Model

- E.F. Codd develops the relational model. Relational database is a type of record-based relations.
- Relational database is an attempt to simplify the data structure by making use of tables. Tables are used to represent the data and their relationships. Table is a collection of rows and columns. Tables are also known as relations.
- Records are known as tuples and fields are known as attributes.

- The relational model is called as record based model because the database is structured in fixed format records of different types. A record is consists of fields or attributes.
- In the relational model, every record must have a unique identification or key based on the data.
- In following table Stud_ID is the key through which we can identify the record uniquely in the relation. Relational data model is the most widely used record-based data model.

Tuple →

Stud_ID	Stud_Name	DOB
101	Prajakta	03/03/1995
102	Rakesh	13/01/1996
103	Rahul	16/08/1995

2. Hierarchical Model

- A data model in which the data is organized into a tree structure is known as hierarchical data model.
- Hierarchical data model structure contains parent-child relationship where root of the tree is a parent which then branches into its children. It is another type of record based data model.
- The data is stored in the form of records. These records are connected to one another.
- A record is a collection of fields; each field contains only one value. The hierarchical data models represent data according to its hierarchy.

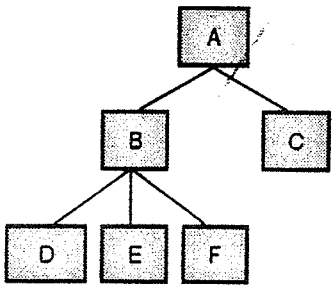


Fig. 2–Q. 1.2 : Hierarchical Model

3. Network Database Model

- It is extended type of hierarchical data model. This data model is also represented as hierarchical, but any child in the tree can have multiple parents.
- In network data model there is no need of parent child association. There is no downward tree structure.

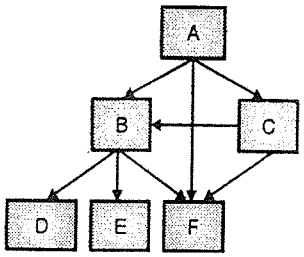


Fig. 3–Q. 1.2 : Network Model

- It is the flexible way of representing the objects and their relationship. A network data model allows multiple records linked in the same file.
- Basically, network database model forms a network like structure between the entities.

4. Entity Relationship (E-R) Model

- This model describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

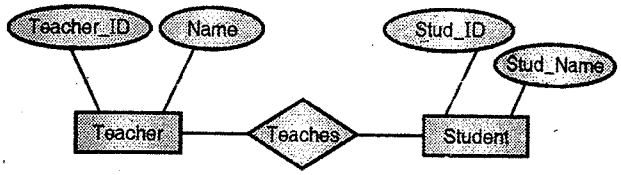


Fig. 4–Q. 1.2 : ER Model

5. Object Oriented Database Model

- Object oriented data model is nothing but the collection of objects or elements with its methods.
- Now a days all advanced programming languages supports the concepts of Object Oriented Programming.
- This model is based on the Object-oriented paradigm. This paradigm is defined for the database.
- It is extension to E-R model with integration of concepts like object, method and encapsulation.
- Applications of data modelling contain the key concepts of object-oriented.

Q. 1.3 Explain the distinction among the terms primary key, candidate key, and super key. **(5 Marks)**

Ans. :

1. Primary Key

- Primary key uniquely identify each entity in the entity set. It must have unique values and cannot hold null values. Let, R be a relationship set having entity sets E1,E2,....En. Consider primary key (Ei) denotes the set of attributes that forms the primary key for entity ser Ei. The set of attributes associated with the relationship set R is responsible for composition of primary key for that relationship set.
- **Example :** In Bank database, the account_number entity should be primary key. Because this field cannot be kept NULL as well as no account_number should be repeated.

2. Super Key

- This key is formed by combining more than one attributes for the purpose of uniquely identifying entities.
- **Example :** In student database having attributes Student_reg_id, Student_roll_no, Student_name, Address, Contact_no.
  - The Super keys are :
    - {Student_reg_id}
    - {Student_roll_no}
    - { Student_reg_id, Student_roll_no }
    - { Student_reg_id, Student_name }
    - { Student_roll_no, Student_name }
    - {Student_reg_id, Student_roll_no, Student_name }
    - It means super key can be any combination of attributes, so that identifying the record becomes easier.

3. Candidate Key

- Candidate key is formed by collection of attributes which hold unique values. A super key without redundant values is known as candidate key. Candidate keys are selected from the set of super keys.
- Candidate key are also known as minimal super key having uniqueness property. The attribute which do not contain duplicate value, may be a candidate key.
- **Example :** In student database with attributes Student_reg_id, Student_roll_no, Student_name, Address, Contact_no.
  - The Candidate keys are :
    - {Student_reg_id}
    - {Student_roll_no}
    - {Student_reg_id, Student_roll_no}
    - It means candidate key can be any combination of key attributes, so that identifying the record from the table becomes easier.

Q. 1.4 Draw an ER-Diagram for online Book Shop which should consist of entity set, attribute, relationship, mapping cardinality and keys, it will maintain information about all Customers, books, book author, publisher, billing etc. **(5 Marks)**

Ans. :

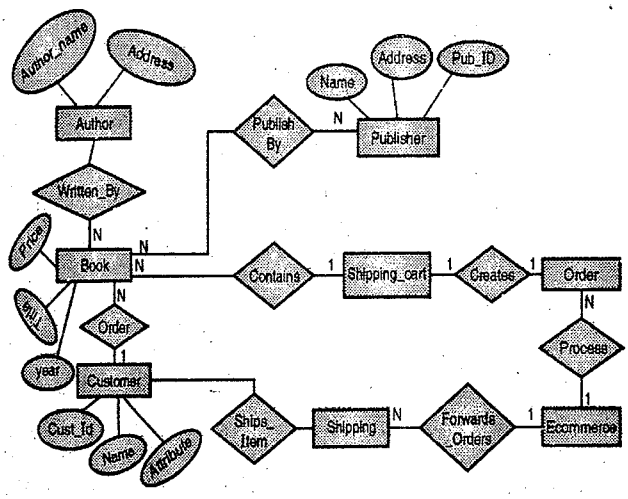


Fig. 1-Q. 1.4

Q. 1.5 What is meant by Mapping Cardinality ? Explain different types of Cardinalities for a binary relationship with example. **(5 Marks)**

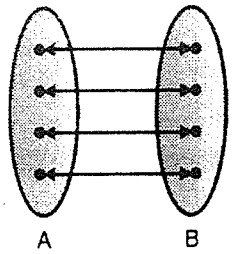
Ans. :

Mapping Cardinalities

- Mapping cardinality expresses the number of entities to which another entity can be associated with in a relationship-set.
- It defines the relationship between two entities via a relationship set. Consider in a binary relationship set R (relation) between entity of set A and set B. For this relationship mapping cardinalities are as Follows :

One to One

- An entity of entity-set A can be associated with at most one entity of entity-set B and vice versa that means an entity in entity-set B can be associated with at most one entity of entity-set A.



(a)

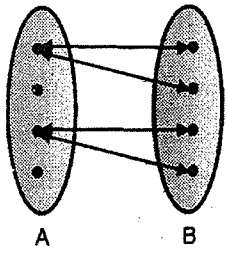
Example : In following example one state have only one capital.

(b)

Fig. 1-Q. 1.5

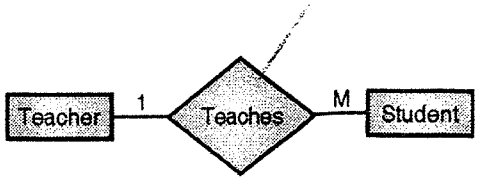
One to Many

- In this type an entity in set A is associated with many other entities in set B.
- But an entity in entity set B can be associated with maximum of one entity in entity set A.



(a)

Example : In following example one teacher can teach many students

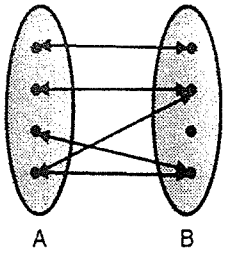


(b)

Fig. 2-Q. 1.5

Many to One

- In this type an entity in set A is associated with at most one entity in set B. And an entity in set B can be associated with number of entities in set A.



(a)

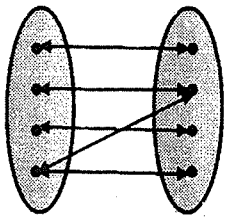
Example : In following example many students can enroll in one school.

(b)

Fig. 3-Q. 1.5

Many to Many

- In this type any entity in entity set A is associated with number of entities in entity set B.



(a)

- An entity in entity set B is associated with number of entities in set entity A.

Example : Many students learn many subjects.

(b)

Fig. 4-Q. 1.5

The appropriate mapping cardinality for a particular relationship set is depending upon the real world situation to which the relationship set is modeling.

Q. 1.6 Explain in detail the different levels of abstraction. **(5 Marks)**

Ans. :

There are three levels of abstraction :

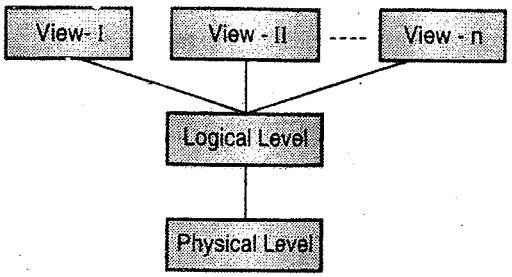


Fig. 1-Q. 1.6

1. Physical level

- In data abstraction Physical level is the lowest level. This level describes how the data is actually stored in the physical memory.
- The physical memory may be hard disks, magnetic tapes, etc. In Physical level the methods like hashing are used for organization purpose.
- Developer would know the requirement, size and accessing frequency of the records clearly in this level which makes easy to design this level.

2. Logical level

- This is the next higher level of abstraction which is used to describe what data the database stores, and what relationships exist in between the data items. The logical level thus describes an entire database in terms of a small number of relatively simple structures.
- Although implementation of the simple structures at the logical level may involve complex physical level structures, the user of the logical level does not need to be aware of this complexity. This is considered as physical data independence.

3. View level

- It is the highest level of data abstraction. This level describes the user interaction with database system. In the logical level, simple structures are used but still complexity remains because in the large database various type of information is stored.
- Many users are not aware of technical details of the system, and also they need not to access whole information from the database. Hence it is necessary to provide a simple and short interface for such users as per their requirements. Multiple views can be created for same database for multiple users.

Q. 1.7 List significant difference between File Processing and DBMS. **(5 Marks)**

Ans. :

Sr. No.	File Processing System	Database Management System
1.	Duplicate data may exist in multiple files which lead to data redundancy.	The data is integrated into a single database which avoids data redundancy.
2.	Data inconsistency occurs when data is not updated in all the files simultaneously.	The data consistency is obtained by controlling the data redundancy
3.	It is difficult to share data in traditional file system.	In DBMS, data can be easily shared by different applications.
4.	In the files, data is stored in specific format. If the format of any of the file is changed, then we have to make changes in program which processes the file.	In DBMS we can completely separate the data structure of database and programs or applications which are used to access the data.
5.	The Traditional File System does not have centralized data control; the data is de-centralized or distributed.	The DBMS provides centralized data storage. Hence keeping control on data is very much easy.

Q. 1.8 What is the purpose of database system ? **(5 Marks)**

Ans. :

- Programming languages like Java, .Net are used to develop customized software's. Every software or application has its data to be stored permanently.
- Programming languages cannot store data permanently. For this purpose we have to use the Database Management System. The DBMS plays a significant role in storing and managing data.
- In an application we store data in DBMS and for operations like insertion, modification or deletion we write code in programming languages i.e. software is usually created with the help of both Programming language and Database.

- When the application is executed on client side, the client or user interacts with interface of application which is created in programming language.
- The database always remains backside and do not come in front of the user. Hence the database is known as backend while programming language is termed as frontend.
- To understand the purpose or need of database system, we need to study the previous option to store data which is called as File Processing System.

Q. 1.9 Explain Advantages of Database Management System. **(5 Marks)**

Ans. :

1. Data Consistency

- The data consistency is obtained by controlling the data redundancy, If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users.
- For example if there is change in designation of employee, then the changes are made in single centralized file which is available to all the users.

2. Sharing of Data

- In DBMS, data can be easily shared by different applications. The database administrator manages the data and gives rights to users to access the data.
- Multiple users can be authorized to access the same data simultaneously. The remote users can also share same data.

3. Data Independence

- In DBMS we can completely separate the data structure of database and programs or applications which are used to access the data.

4. Data Control

- The DBMS provides centralized data storage. Hence keeping control on data is very much easy as compared to Traditional File Processing System.
- As data is common for all the application, no possibility of any confusion or complication.

Q. 1.10 Explain disadvantages of Database Management System. **(5 Marks)**

Ans. :

1. Increased Costs

- To install Database Systems, we require standard software and hardware. Also to handle the Database System, highly skilled personnel are required.
- The cost of maintaining the software, hardware, and personnel required to operate and manage the database system is more.

2. Complexity

- Sometimes because of higher functionality expectations, the design of Database may become very complex.
- To utilize such database with complete efficiency, all the stakeholders like database designers, developers, database administrators and end-users must understand the functionality.

3. Size

- The DBMS becomes extremely large piece of software because of the complexity of functionality occupying large amount of disk space and requiring substantial amounts of memory to run efficiently.

4. Frequent Upgrade/Replacement Cycles

- New functionalities are often added into DBMS by their vendors. These new features often come bundled in new upgraded versions of the same software.
- Sometimes these versions require hardware upgrades which increases expenses. Also work to train database users and administrators to properly use and manage the new features get increased.

5. Higher Impact of a Failure

- The DBMS is placed at centralized location which increases the vulnerability of the system.
- That means the DBMS may get attacked and harmed. Since all users and applications rely on the centralized database, the failure of any component can bring operations to a halt.

Q. 1.11 What is entity explain with its types ? **(5 Marks)**

Ans. :

- An **entity** is nothing but a thing having its own properties. These properties helps to differentiate the object (entity) from other objects.

There are two types of entities in Database management system.

1. Strong Entity or Regular Entity

- If an entity having it's own key attribute specified then it is a strong entity. Key attribute is used to indentify that entity uniquely among set of entities in entity-set.

  Example : In a parent/child relationship, a parent is considered as a strong entity.

- Strong entity is denoted by a single rectangle.
- The relation between two strong entities is denoted by a single diamond simply called relationship.

2. Weak Entity

- The entity which does not have any key attribute is known as weak entity. The weak entity has a partial discriminator key. Weak entity depends on the strong entity for its existence. Weak entity is denoted with the double rectangle.

  Example : In a parent/child relationship, a child is considered as a weak entity which is completely depends upon the strong entity 'parent'.

Q. 1.12 What is attribute explain with its types ? **(5 Marks)**

Ans. :

- An attribute is a characteristic of an entity. Entities are represented by means of their attributes. All attributes have their own specific values. For example, an employee entity may have Employee_ID, emp_name, salary as attributes.
- There are five different types of attributes in Database Management System :

1. Single-valued Attribute

- A single-valued attribute is the attribute which can hold a single value for the single entity.
- Example : In the entity student, student_name is the single-valued attribute since a student have a single value for name attribute.

2. Multi-valued Attribute

- A multi-valued attribute is the attribute which can hold multiple values for the single entity.
- Example : In the entity student, the attribute student_contact no could be considered a multi-value attribute since a student could have multiple contact numbers.

3. Simple Attribute

- An attribute whose value cannot be further divided is known as simple attribute. That means it is atomic in nature.

  Example : In the entity student, the attribute student_age cannot be divided. Therefore student_age is the simple attribute of student entity.

4. Composite Attribute

- The composite attributes are the attributes which can be further divided into sub parts. These sub parts represent the basic entities with their independent meaning.

  Example : In the entity student, student_name is the composite attribute, we can divide this attribute in three different sub parts: First_name, Middle_name and Last_name.

5. Derived Attribute

- The attribute which is not physically exist in database, but its value can be calculated from the other present attributes is known as derived attribute.

  Example : In the entity student, we can calculate the average age of students. This average age is not physically present in the database but it can be derived from the attribute student_age;

Q. 1.13 Explain the concept of specialization.

Ans. :

- In an entity set, sometimes further sub grouping is also possible depending upon certain characteristics. For example, a subset may have some attributes which are not shared by other subset in the entity set. In E-R diagram these distinct subsets can be represented by some means.
- Consider an example of entity set person having attributes name, city and street. The entity person may be further classify as follows:
  - customer
  - employee
- Both of these types of person are described by a set of attributes that contains all the attributes of entity set person with some additional attributes of their own.
- For example, in the description of customer entities, we may add new attribute like customer-id, whereas in the description of employee entities we may add new attributes like employee-id and salary.

- This process of creating subgroups within an entity set is called **specialization**. The specialization of entity person helps us to distinguish whether a person is an employee or customer depending upon attributes.
- An entity set may be specialized by more than one distinguishing characters. In the above example, job performed by an employee may be a distinguishing character from customer. The employee may be further divided as permanent or temporary employee depending upon some characteristics.

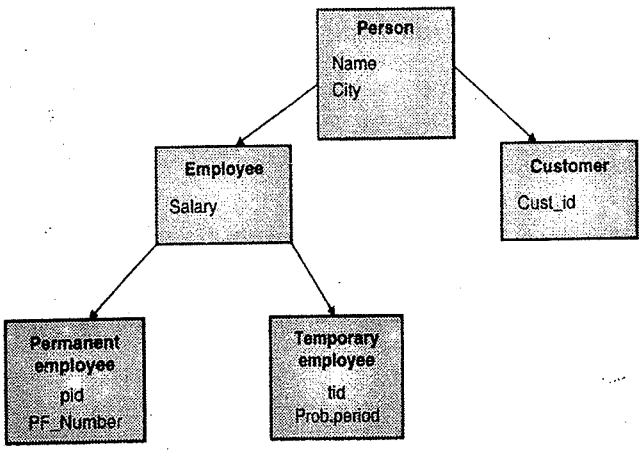


Fig. 1-Q. 1.13 : Specialization

Chapter 2 : SQL and PL/SQL

Q. 2.1 Consider the relational database Supplier (Sid, Sname, address), Parts (Pid, Pname, color), Catalog (sid, pid, cost)

Write SQL queries for the following requirements :

i) Find name of all parts whose color is green.

ii) Find names of suppliers who supply some red parts.

iii) Find names of all parts whose cost is more than Rs.25. **(5 Marks)**

Ans. :

Consider the following 3 tables

```
mysql> select *from supplier;
+-----+-------+---------+
| sid | sname | address |
+-----+-------+---------+
| s1  | abc   | Pune    |
| s2  | pqr   | Mumbai  |
| s3  | xyz   | Nasik   |
+-----+-------+---------+
3 rows in set (0.08 sec)

mysql> select *from parts;
+-----+-------+-------+
| pid | pname | color |
+-----+-------+-------+
| p1  | prod1 | red   |
| p2  | prod2 | green |
| p3  | prod3 | blue  |
+-----+-------+-------+
3 rows in set (0.05 sec)
```

```
mysql> select *from catalog;
+------+------+------+
| sid  | pid  | cost |
+------+------+------+
| s1   | p1   |   30 |
| s2   | p3   |   10 |
| s3   | p2   |   40 |
+------+------+------+
3 rows in set (0.00 sec)
```

i) Find name of all parts whose color is green.

```
Select * from parts where color = green;
```

Output

```
mysql> select* from  parts where color='green';
+-----+-------+-------+
| pid | pname | color |
+-----+-------+-------+
| p2  | prod2 | green |
+-----+-------+-------+
1 row in set (0.02 sec)
```

ii) Find names of suppliers who supply some red parts.

```
Select s.sname, p.color from supplier s, parts p, catalog c
where s.sid = c.sid and p.pid = c.pid and p.color ='red';
```

Output

```
mysql> Select s.sname, p.color from supplier s,
 c.sid and p.pid = c.pid and p.color ='red';
+-------+-------+
| sname | color |
+-------+-------+
| abc   | red   |
+-------+-------+
1 row in set (0.02 sec)
```

iii) Find names of all parts whose cost is more than Rs.25

```
select p.pname, c.cost from parts p, catalog c where p.pid =
c.pid and c.cost > 25;
```

Output

```
mysql> select p.pname, c.cost from parts p, catalog c
    -> where p.pid = c.pid and c.cost > 25;
+-------+------+
| pname | cost |
+-------+------+
| prod1 |   30 |
| prod2 |   40 |
+-------+------+
2 rows in set (0.05 sec)

mysql>
```

Q. 2.2 Consider Following Relational Tables :

Student (Roll-no, name, address)

Subject (Sub_code,Sub-name)

Marks (Roll-no, Sub-code, marks)

Solve following queries using SQL

(i) Find out average marks of each student, along with the name of student.

(ii) Find how many students have failed in the subject "DBMS". **(5 Marks)**

Ans. :

```
Create table student(roll_no int(3) , name varchar(10),address
varchar(20), primary key(roll_no));
Create table subject(sub_code varchar(10),sub_name
varchar(10), primary key(sub_code));
Create table marks(roll_no int(3) , sub_code
varchar(10),marks int(3), foreign key(roll_no) references
student(roll_no), foreign key(sub_code) references
subject(sub_code));
```

Consider following tables

```
mysql> select *from manages;
+-------+-------+
| pname | mname |
+-------+-------+
| xy    | aaa   |
| pqr   | bbb   |
| abc   | ccc   |
+-------+-------+
3 rows in set (0.00 sec)

mysql>
```

```
mysql> select *from subject;
+----------+----------+
| sub_code | sub_name |
+----------+----------+
| c        | C Prog   |
| jv       | Java     |
| or       | Oracle   |
+----------+----------+
3 rows in set (0.00 sec)
```

```
mysql> select *from marks;
+---------+----------+-------+
| roll_no | sub_code | marks |
+---------+----------+-------+
|       1 | or       |    90 |
|       2 | c        |    80 |
|       3 | jv       |    78 |
+---------+----------+-------+
3 rows in set (0.00 sec)
```

(i) Find out average marks of each student, along with the name of student.

```
Select st.name,su.sub_code, m.marks from student st,subject
su,marks m where st.roll_no = m.roll_no and
su.sub_code=m. sub_code;
```

Output

```
+------+----------+-------+
| name | sub_code | marks |
+------+----------+-------+
| abc  | or       |    90 |
| pqr  | c        |    80 |
| xyz  | jv       |    78 |
+------+----------+-------+
3 rows in set (0.02 sec)
```

(ii) Find how many students have failed in the subject "Java".

```
Select st.name, su.sub_name, m.marks from student st,subject
su,marks m where st.roll_no = m.roll_no and
su.sub_code=m. sub_code and m.marks > 40 and
su.sub_name='Java';
```

Output : Empty Set as no student is there who failed in Java.

Q. 2.3 Explain Stored Procedures. **(5 Marks)**

Ans. :

Stored procedure is a group of SQL statements which can be executed repeatedly. It allow for variable declarations, flow control and other useful programming techniques. Parameters are the

important part of procedures. The parameters make the stored procedure more flexible and useful.

Consider the table

```
mysql> select *from  student;
+------+--------+------------+-------+
| rno  | name   | dob        | class |
+------+--------+------------+-------+
|    2 | aa     | 1990-11-21 | NULL  |
|    3 | priya  | 1989-10-30 | NULL  |
|    4 | asd    | 2000-02-02 | NULL  |
|    5 | ggg    | 0000-00-00 | NULL  |
+------+--------+------------+-------+
4 rows in set (0.00 sec)

mysql>
```

Syntax

```
DELIMITER //
CREATE procedure procedure _name ([parameter(s)])
STATEMENTS
DELIMITER //
```

IN Parameter : Accepts value when procedure get called.

Example : Create a procedure which should accept rollno as parameter and display the record.

```
DELIMITER //
CREATE PROCEDURE display(IN r integer (3))
BEGIN
SELECT * FROM students WHERE rno = r;
END //
DELIMITER ;
```

Now the procedure can be called as

```
CALL display(3);
```

Output

```
+------+-------+------------+-------+
| rno  | name  | dob        | class |
+------+-------+------------+-------+
|    3 | priya | 1989-10-30 | NULL  |
+------+-------+------------+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.06 sec)
```

Out Parameter : The value of an OUT parameter can be changed inside the stored procedure. The changed value is passed back to the calling program. The initial value of OUT parameter cannot be accessed by the procedure.

Example

```
DELIMITER $$
CREATE PROCEDURE display1(IN r INT,OUT nm
VARCHAR(25))
BEGIN
select name into nm from students where rno = r;
END $$
DELIMITER ;
```

Now the procedure can be called as

```
CALL display1(3 ,@n);
```

Here n is the OUT parameter which stores the name of student having rno 3;

Then execute the command

```
Select @n;
```

Output

```
+-------+
| @n    |
+-------+
| priya |
+-------+
1 row in set (0.00 sec)

mysql>
```

Q. 2.4 What is cursor? Explain various types of Cursor. **(10 Marks)**

Ans. :

- **Cursor** is used to traverse in the database to access the records one by one. For example if we want to calculate the average of marks of all the students, then we can retrieve marks of every student and add in the total_marks. Cursor is just like loop concept used to traverse to every row and manipulate data.
- An area of memory(Context) is allocated for the processing of SQL statements. The context area contains information necessary to complete the processing, including the number of rows processed by the statement, a pointer to the parsed representation of the statement.
- Cursor is a handle or pointer to the context area.
  - There are two types of cursors
  - (a) Implicit Cursor
  - (b) Explicit cursor

(a) Implicit Cursor

- When there is no explicit cursor, the implicit cursors are created automatically whenever an SQL statement is executed. Programmers cannot control the implicit cursors and the information in it.
- When Data Manipulation statements like insert, update or delete are executed, an implicit cursor is automatically

associated with these statements. In case of insert statement the data is hold by cursor while for delete and update statements, cursor identifies the rows that would be affected.

- Consider following example in which increment of 10% is given to all the employees. Here an implicit cursor is created to identify the set of rows in the table which would be affected by the update.

```
UPDATE employee
 SET salary = salary * 0.1;
```

(b) Explicit Cursor

An explicit cursor is the one in which the cursor name is explicitly assigned to the select statement. Processing an explicit cursor involves following three steps.

(1) **Open** : The Open statement executes the select statement. Positions the cursor at the first row.

(2) **Fetch** : The Fetch statement retrieves the current row and advances the cursor to the next row for processing.

(3) **Close** : After processing of last row the cursor is disabled with the help of Close statement.

Example : Consider the table city

```
mysql> select *from city;
+-----+-----------+----------+
| id  | city_name | state_id |
+-----+-----------+----------+
| 101 | Pune      |        1 |
| 102 | Mumbai    |        1 |
| 103 | Nasik     |        1 |
+-----+-----------+----------+
```

```
DELIMITER //
CREATE PROCEDURE cname(
 IN in_state_id INT
)
BEGIN
 DECLARE record_not_found INTEGER DEFAULT 0;
 DECLARE mycity_name VARCHAR(255)
DEFAULT "";
 DECLARE mysql_cursor CURSOR FOR
   SELECT city_name FROM city
   WHERE state_id = in_state_id;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET
record_not_found = 1;
 OPEN mysql_cursor;
 cities_loop: LOOP
 FETCH mysql_cursor INTO mycity_name;
  IF record_not_found THEN
   LEAVE cities_loop;
  END IF;
 SELECT mycity_name;
END LOOP cities_loop;
CLOSE mysql_cursor;
END //
DELIMITER ;
```

To call the procedure give query

```
CALL cname(1);
1 is the state id.
```

Q. 2.5 What are the advantages of SQL? **(5 Marks)**

Ans. :

There are various advantages of SQL

- **High Speed :** SQL Queries can be used to retrieve large amounts of records quickly and efficiently from a database.
- **Portable :** SQL can be run on any platform. Also it can be executed on PCs, laptops, servers and even mobile phones. It runs in local systems, intranet and internet. Databases using SQL can be moved from a device to another without any problems.
- **Well Defined Standards Exist :** SQL databases use long-established standard, which is being adopted by ANSI & ISO. Whereas Non-SQL databases do not adhere to any clear standard.
- **Supports object based programming :** SQL supports various object oriented programming concepts which makes it powerful.
- **Used with all DBMS systems with any vendor :** SQL is used by all the vendors who develop DBMS.
- **No Coding Required :** Using standard SQL it is easier to manage database systems without writing large amount of code.
- **Used for relational databases :** SQL is widely used for number of relational databases.

- **Easy to learn and understand :** SQL mainly consists of English words and hence it is easy to learn and understand the SQL queries.

Q. 2.6 Explain any four data types of SQL. **(5 Marks)**

Ans. :

1. BOOLEAN

- The BOOLEAN data type can accepts value either TRUE or FALSE. No need to declare size while declaring the BOOLEAN data type.
- TRUE or FALSE are case insensitive. If you attempt to assign any other value to a BOOLEAN data type, an error gets raised.

Examples : TRUE, true, True, False

2. FLOAT (p)

- The FLOAT data type accepts approximate numeric values, for which you may define a precision up to a maximum of 64. The default precision is 64 if not declared.

Examples : FLOAT(8)

12345678, 1.2, 123.45678, -12345678, -1.2, -123.45678

3. INTEGER or INT

- The INTEGER data type is used to accept numeric values with a default scale as zero. It stores any integer value between the range 2^ -31 and 2^31 -1. Attempting to assign values outside this range causes an error.
- If you assign a numeric value with a precision and scale to an INTEGER data type, the scale portion truncates, without rounding.

Examples

– 345, 0, 4532.98 (digits to the right of the decimal point are truncated), 167

4. DATE

- The DATE data type accepts date type of values. No need to assign size while declaring a DATE data type. Date values should be specified in the form: YYYY-MM-DD.
- The value of month must be between 1 and 12, value of day should be between 1 and 31 depending on the month and value of year should be between 0 and 9999. The values should be enclosed in single quotes, preceded by the keyword DATE.

Examples : DATE '1999-01-01'

DATE '2000-2-2'

Q. 2.7 Enlist literals in SQL. Explain any three.

Ans. :

- SQL Supports following types of literals.

(1) Numeric Literals

(2) Character Literals

(3) String Literals

(4) Date Literals

(5) Time Literals

(1) Numeric Literals

- Numeric literals are the sequence of digits proceeded by an optional sign (+ / –) and with an optional decimal point. The Numeric Literals are further classified into two categories :
- **Integer Literals :** These are the whole numbers assigned as data values. An integer can store a maximum of 38 digits of precision.

For example : 100, + 7, – 8 etc.

- **Number or Floating Point Literals :** These are the numbers with decimal point assigned as data values. For example : 10.2, +7.3, -8.2 etc.

(2) Character Literals

- Character literal contains single character enclosed in single quotation marks.

For example : 'A', '%', '9', 'z', '('

(3) Date Literals

- These literal are the date type of values in the ANSI date format 'YYYY-MM-DD' or the default date format specified in the application via the 'set date format' operation.
- The date literal is enclosed in single quotation marks.

For Example : '2017-03-18'

- If the am/pm indicator is excluded, it is assumed that the time is in 24-hour format.
- The date literal is enclosed in single quotation marks.

'11:15', '8:30 am', '06:25:15 pm', '17:00'

Q. 2.8 Explain types of index. **(5 Marks)**

Ans. :

1. Single Column Index

This is index is created on a single column of a table.

Syntax

```
CREATE INDEX index_name
ON table_name (column_name)
```

Example

```
CREATE INDEX ind1
on student(stud_name);
```

2. Composite Index

Sometimes duplicate records may available in columns. In such case the composite indexing is better option to index the data. This index is created on a multiple columns of a table.

Syntax

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

Example

```
CREATE INDEX ind2 on student(stud_name , marks);
```

3. Unique Index

A unique index does not allow any duplicate values to be inserted into the table.

Syntax

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

Example

```
CREATE UNIQUE INDEX ind3
on student(stud_name);
```

4. Implicit Index

Implicit indexes are indexes that are automatically created by the database server when an object is created. Such indexes are created for primary key and unique constraints.

Displaying Index : To display index information regarding table following query is used.

Syntax

```
Show index from table_name;
```

Example

```
Show index from student;
```

Q. 2.9 Enlist set operation in SQL. Explain any two. **(5 Marks)**

Ans. : The different Set Operators are as follows

(i) Union (ii) Union All

(iii) Intersect (iv) Minus

(i) Union : The union operator returns all distinct rows selected by either query

Syntax

```
Select column_name from table_1
Union
Select column_name from table_2
```

Example

```
Select DeptNo from emp
union
select Deptno from dept
```

Output

10
20
30
40

(ii) Union All : The Union All operator returns all rows selected by either query including duplicates.

Syntax

```
Select column_name from table_1
Union all
Select column_name from table_2
```

Example

```
Select DeptNo from emp
Union all
select Deptno from dept
```

Output

10
20
30
20
10
10
20
30
40

Q. 2.10 Explain Like Predicate with example. **(5 Marks)**

Ans. :

Like operator determines whether a specific character string matches the given pattern or not. In the pattern we can use regular characters and wildcard characters. In this pattern matching, it is necessary that regular characters must exactly match the characters specified in the character string. However, for the wildcard characters arbitrary fragment matching of the character string is done. The use of wildcard characters makes the LIKE operator more flexible.

Example : Display records of employee whose names starts with letter 'J'

Query

```
select * from emp
where ename like 'J%';
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7900	JAMES	CLERK	7698	12/03/1981	950		30

Example

Display records of employee whose names ends with letter 'N'

Query

```
select * from emp
where ename like '%N';
```

Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30

Q. 2.11 Write note on Aggregate Functions. **(5 Marks)**

Ans. :

Aggregate functions perform a calculation on a set of values and return a single value. Usually these functions ignore NULL values(except for COUNT).

There are different types of aggregate functions :

(i) Min (ii) Max (iii) Sum

(iv) Avg (v) Count

Consider the following table:

Eno	Ename	Job	Sal
102	Ajay	Manager	18000
104	Bharati	Manager	17000
103	Dinesh	Clerk	10000
105	Prajakta	Salesman	13000
101	Susheel	Clerk	12000

(i) **Min() :** This function returns smallest value from specified column of the table.

Query

```
Select min(sal) from emp;
```

Output

```
1000
```

(ii) **Max() :** This function returns greatest value from specified column of the table.

Query

```
Select max(sal) from emp;
```

Output

```
18000
```

(iii) **Sum() :** This function returns sum of all the values of specified column of the table.

Query

```
Select sum(sal) from emp;
```

Output

```
70000
```

(iv) **Avg() :** This function returns average of all the values of specified column of the table.

Query

```
Select avg(sal) from emp;
```

Output

```
14000
```

(v) **Count() :** This function returns total number of values of specified column of the table.

Query

```
Select count(ename) from emp;
```

Output

```
5
```

Q. 2.12 Write a short note on Dynamic SQL.

(5 Marks)

Ans. :

Dynamic SQL refers to SQL code that is generated in an application or from the system tables and then executed against the database to manipulate the data at run time. The SQL code is not stored in the application but rather it is collected depending upon the input given by user.

The dynamic SQL helps to develop powerful applications which allows us to create database objects and manipulate them based on the input provided by the user.

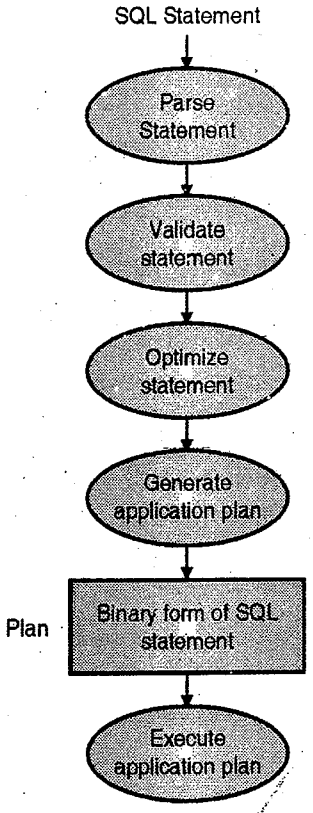


Fig. 1-Q. 2.12

- In dynamic SQL the column_name or Table_name are not known at compile time so the DBMS can't able to prepare the SQL statement for execution in advance.
- In this method, when the program is executed, the SQL statement get the table_name or column_name from user and this statement is given to the DBMS for further execution.
- In the Fig. 1-Q. 2.12 we can observe the general execution process of dynamic sql.

An Example of Dynamic SQL statement

```
mysql> PREPARE stmt FROM
  -> 'select count(*)
  -> from information_schema.schemata
  -> where schema_name = ? or schema_name = ?';

Query OK, 0 rows affected (0.00 sec)
Statement prepared
mysql> EXECUTE stmt
  -> USING @schema1,@schema2
```

Output

```
+---------+
| count(*) |
+---------+
|     2    |
+---------+
1 row in set (0.00 sec)

mysql> DEALLOCATE PREPARE stmt;
```

Chapter 3 : Relational Database Design

Q. 3.1 Explain Characteristics of Relational Database.

(5 Marks)

Ans. :

The characteristics of Relational database systems are as follows :

- This model is called as Relational Model by Dr. Codd because the data is stored in the tables which are having relationships in between them.
- The whole data of the system is represented as systematic arrangement of data into rows and columns, called as relation or table.
- A table is also form in two-dimensional structure.
- At any given row/column position means in every cell in the relation there is one and only one value which is known as scalar value.
- Column represents attribute, and each column has a distinct name.
- All values entered in the columns are of the same data format.
- Implements the concept of closure means all operations are performed on an entire relation and result is an entire relation.
- It supports the operations like data definition, data manipulation and transaction management.

Q. 3.2 Explain advantages of Relational Database. **(5 Marks)**

Ans. :

(a) **Ease of use :** The system which is managed in the form of tables consisting of rows and columns is much easier to understand.

(b) **Flexibility :** The information from multiple tables can be retrieved easily at a time by joining the tables. Also changes can be done easily by using different operators.

(c) **Security :** The different users can have different levels of access to data based on their roles. In the college database, students will have access to their own data only, while their teachers will have access to data of all the students to whom they are teaching. Class teacher will be able to see the reports of all the students in that class, but not other classes. The principal will have access to entire data.

(d) **Data Independence :** In Relational system we can completely separate the data structure of database and programs or applications which are used to access the data. This is called as data independence. If any changes are made in structure of database then there is no need to make changes in the programs. For example you can modify the size or data type of a data items (fields of a database table) without making any change in application.

(e) **Data Manipulation Language :** In the relational database approach it is easy to respond query by means of a language like SQL based on relational algebra and relational calculus. For data organized in other structure the query language either becomes complex or extremely limited in its capabilities.

Q. 3.3 Differentiate between primary Key constraint & Foreign key constraint. **(5 Marks)**

Ans. :

Sr. No.	Primary Key	Foreign Key
1.	Primary key uniquely identify a record in the table.	Foreign key is a field in the table that is primary key in another table.
2.	Primary Key can't accept null values.	Foreign key can accept null values.
3.	By default, Primary key is clustered index and data in the database table is physically organized in the sequence of clustered index.	Foreign key do not automatically create an index, clustered or non-clustered. You can manually create an index on foreign key.
4.	We can have only one Primary key in a table.	We can have more than one foreign key in a table.

Q. 3.4 Enlist Features of Good Relational Designs. **(5 Marks)**

Ans. :

- It should be able to distribute the information into different tables to reduce redundancy of data.
- It should provide easy access to the data available in multiple tables.
- It should take care of accuracy and integrity of information.
- It should provide functionalities for data processing and reporting needs.
- Data entry, updates and deletions should be efficient.
- Data retrieval, summarization and reporting should also be efficient.
- The database design must be self-documenting since much of the information is stored in the database rather than in the application.
- To understand the features of a good relational design we will see an example.

Q. 3.5 Explain third normal forms with suitable example. **(5 Marks)**

Ans. :

- A database design is said to be in 3NF if both the following conditions are satisfied by it
- Initially the design must be in 2NF.

- No non-prime attribute should be transitively dependent on prime key attribute.
- For any non-trivial functional dependency, $X \rightarrow A$
- Either X is a superkey or A is prime attribute.
- Consider the Table 1-Q. 3.5.

Table 1-Q. 3.5 : STUDENT_DETAILS

STUD_ID	STU_NAME	CITY	ZIP
S101	Kunal	Pune	411037
S102	Radhika	Nasik	422001
S103	Kiran	Mumbai	400016
S104	Jay	Nagpur	440001

- In Table 1-Q. 3.5 the STUD_ID is the only one primary key. Here the data of city can be retrieved through either STUD_ID or ZIP code. But the CITY is not superkey as well as nor the CITY is prime attribute.
- Here the CITY is depends upon ZIP and ZIP is depends upon STUD_ID

  Stud_id → zip → city.
- This relationship is known as transitive dependency.
- We have to remove this transitive dependency by implementing Third Normal Form. For this purpose we have to split the STUDENT_DETAILS table in two different tables say STUDENT_DATA and CITY.

STUDENT_DATA

STUD_ID	STU_NAME	CITY
S101	Kunal	Banking System
S102	Radhika	Library Management
S103	Kiran	Speech to Text Converter
S104	Jay	ATM

CITY

ZIP	CITY
411037	Pune
422001	Nasik
400016	Mumbai
440001	Nagpur

- Now the database design is converted into Third Normal Form. Here the basically design is already in Second Normal Form and No non-prime attribute is transitively dependent on prime key attribute.

Q. 3.6 Differentiate between 3NF & BCNF. How it is stronger than 3NF ? **(5 Marks)**

Ans. :

Sr. No.	3NF	BCNF
1.	It stands for Third Normal Form.	It stands for Boyce-Codd Normal Form.
2.	A database design should be already in 2NF to convert in 3NF.	A database design should be already in 3NF to convert in BCNF.
3.	Rule : For any non-trivial functional dependency, $X \rightarrow A$ Either X is a superkey or A is prime attribute	Rule : For any non-trivial functional dependency, $X \rightarrow A$ X must be a super-key.
4.	3NF is weaker than BCNF.	BCNF is stronger than 3NF.
5.	3NF cannot catch all the anomalies.	BCNF was developed to capture those anomalies that could not be captured by 3NF.
6.	More redundancy.	Less redundancy.

Q. 3.7 Differentiate between 4NF & BCNF. **(5 Marks)**

Ans. :

Sr. No.	4NF	BCNF
1.	It stands for Fourth Normal Form.	It stands for Boyce-Codd Normal Form.

Sr. No.	4NF	BCNF
2.	A database design should be already in 3NF and BCNF to convert in 4NF.	A database design should be already in 3NF to convert in BCNF.
3.	No multi-valued dependencies exist in the tables.	Multi-valued dependencies exist in the tables.
4.	The 4NF is more desirable than BCNF because it avoid repetition of data.	The BCNF is less desirable than BCNF because it does not avoid repetition of data.
5.	The decomposition in 4NF does not lead to loss of information when lossless join decomposition is used.	This is little bit difficult in BCNF.
6.	Redundancy is less.	Redundancy is more.

Q. 3.8 Write short note on temporal data. **(5 Marks)**

Ans. :

- **Temporal data** is the data to which time period is attached to it. This time period indicates that when that data is valid or stored in the database. By attaching a time period to the data, it becomes possible to store different states of the database.
- There are forms of temporal databases which depend upon the two different notions of time - valid time and transaction time.
- **Historical database** stores data with respect to valid time.
- **Rollback database** stores data with respect to transaction time.
- **Bitemporal database** stores data with respect to both valid time and transaction time.
- Usually in the commercial DBMS only a single state of the real world is stored, in general the most recent state. Such databases are known as **snapshot databases**.
- In the context of valid time and transaction time, the snapshot database is denoted in the Fig. 1-Q. 3.8.

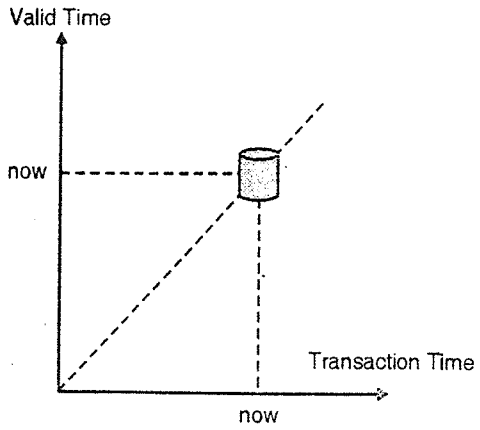


Fig. 1-Q. 3.8

Q. 3.9 What is decomposition explain with example ? **(5 Marks)**

Ans. :

- Breaking a relation into two or more relations. This process is called as **decomposition**.
- Let, R be a relation schema.
- A set of relation schemas {R1, R2,..., Rn} is a decomposition of R if

$$R = R1 \cup R2 \cup .....\cup Rn$$

Example

- Let us know how to break the relation into more than one relation.
- Consider CLASSINFO relation having COURSE_ID, COURSE_NAME, STUD_NAME, SUB_ID, and SUB_NAME.

Table 1-Q. 3.9 : CLASSINFO

COURSE_ID	COURSE_NAME	STUD_NAME	SUB_ID	SUB_NAME
C_101	F. E.	Nishant	S_112	C
C_102	S. E.	Prashant	S_212	Java
C_102	S. E.	Prashant	S_222	DBMS
C_103	T. E	Sanvi	S_311	Ad. Java
C_101	F. E	Sanvi	S_115	CPP

This relation can be decomposed into two relations as follows:

i) COURSE_ STUDENT(COURSE_ID, COURSE_NAME, STUD_NAME)

ii) SUBJECT_ STUDENT(SUB_ID, SUB_NAME, STUD_NAME)

Table 2-Q. 3.9 : COURSE_STUDENT

COURSE_ID	COURSE_NAME	STUD_NAME
C_101	F. E.	Nishant
C_102	S. E.	Prashant
C_103	T. E	Sanvi
C_101	F. E	Sanvi

Table 3-Q. 3.9 : SUBJECT_STUDENT

SUB_ID	SUB_NAME	STUD_NAME
S_112	C	Nishant
S_212	Java	Prashant
S_222	DBMS	Prashant
S_311	VB	Sanvi
S_115	CPP	Sanvi

Q. 3.10 Write short note on Single-valued Functional Dependency. **(5 Marks)**

Ans. :

- Database is a collection of related information in which information depends on another information.
- The information is either single-valued or multi-valued. For example, the name of the person or his / her date of birth is single valued facts. But the Contact number of a person is a multi-valued attribute.
- A single valued functional dependency is when A is the primary key of an relation (e.g. STUDENT_ID) and B is some single valued attribute of the relation (e.g. STUDENT_NAME).
- Then, A → B must always hold by the relation.

STUDENT

STUDENT_ID	STUDENT_NAME	ADDRESS	DOB	COURSE
1001	Akash	Nagpur	11-2-1980	Oracle
1002	Anuja	Pune	2-10-1982	Java
1003	Anuja	Jalgaon	23-10-1981	PHP
1004	Amruta	Nanded	20-9-1981	DBMS

STUDENT_ID → STUDENT_NAME

1002 Anuja

For every STUDENT ID there should be unique name

(A → B)

Q. 3.11 Short note on : Multivalued Dependency. **(5 Marks)**

Ans. :

- A **multi-valued dependency** exists when a relation R has at least 3 attributes (like A, B and C) and for value of A there is a well defined set of values of B and a well defined set of values of C.
- However, the set of values of B is independent of set C and vice versa.

Example

Table 1-Q. 3.11 : Published_Book

Book_Id	Auther_Name	Price
B_001	Jasmin	1000
B_002	Jasmin	980
B_003	Smita	1500
B_004	Smita	2000

- As shown in Table 1-Q. 3.11, name of particular author is determined by Book_Id attribute.

{Book_Id} ->>{Author_Name}

- Also it is possible to determine the Price of book by Book_Id.

{Book_Id} ->> {Price}

- But it is difficult to determine the price of book using author name or vice versa. So this dependency is called as multi-valued functional dependency or simply multi-valued dependency.

Q. 3.12 Short note on : Properties of Armstrong's axioms. **(5 Marks)**

Ans. :

The Armstrong's Axioms are **sound** :

- When basic functional dependencies F on a relation R are given, then the FDs generated by using Armstrong's axioms will hold by R.
- In other words applying Armstrong axioms on set of FDs which are previously defined will not generate invalid FDs.

- The Armstrong's Axioms are **complete** : When basic functional dependencies F on a relation R are given, then each and every valid Functional dependency on relation R can be found by applying only Armstrong axioms on set of FDs which are previously defined.

Union or Additivity

If in relation R having three sets of attributes X, Y, Z, and the set of functional dependencies : $X \rightarrow Y$ and $X \rightarrow Z$ are hold by relation R, then the functional dependency $X \rightarrow YZ$ also hold by relation R. This rule is called as **union**.

Decomposition or Projectivity

If in relation R having three sets of attributes X, Y, Z and the functional dependency $X \rightarrow YZ$ is hold by relation R then the following sets of Functional dependencies also hold by relation

R: $X \rightarrow Y$ and $X \rightarrow Z$

This inference rule is called as **decomposition rule**.

Pseudo transitivity

If in relation R having four sets of attributes W, X, Y, Z and the functional dependencies $X \rightarrow Y$ and $YW \rightarrow Z$ are hold by relation R, then the relation also hold following functional dependency $XW \rightarrow Z$. This inference rule known **pseudo transitivity**.

Q. 3.13 Write a short note on first normal form. **(5 Marks)**

Ans. :

First normal form

- A **domain** is the set of all unique values which is permitted for an attribute. **Atomic** means that cannot be divided further.
- First Normal Form defines that all the attributes in a relation must have atomic (not further divisible) and single values.
- Consider the table Course_details

Table 1-Q. 3.13 : Course_details

Category_id	Category_name	Languages
C_PRG	Programming	C, VB, Java
C_SCR	Scripting	JavaScript, PHP, HTML

- Table 1-Q. 3.13 is **not in 1NF** as the rule says "each attribute of a table must have atomic (single) values", the attribute 'Languages' contain multiple values which violets the rule of 1NF. To convert this data into First Normal Form, we have to rearrange it in the table.

Category_id	Category_name	Languages
C_PRG	Programming	C
C_PRG	Programming	VB
C_PRG	Programming	Java
C_SCR	Scripting	JavaScript
C_SCR	Scripting	PHP
C_SCR	Scripting	HTML

- Now each attribute contain single values. Hence the database design is in First Normal Form.

Q. 3.14 What is anomalies ? Explain any two types of it. **(5 Marks)**

Ans. :

Anomalies are inconvenient or error-prone situation arising when we process the tables. There are three types of anomalies :

A) Insert Anomaly

- An **Insert Anomaly** occurs when it is not possible to insert certain attributes into the database without the availability of other attributes.
- For example, In College Database System, it is not possible to add entry of any new course unless any student enrolled for it.

Table 1-Q. 3.14 : STUDENT_INFO

STUD_ID	COURSE_ID	STU_NAME	ADDRESS	COURSE
S101	C1	Kunal	Camp	VB
S101	C2	Kunal	Camp	Java
S102	C3	Radhika	Sahkar Nagar	Oracle
S103	C4	Kiran	Tilak Road	C++
S104	C5	Jay	Pune Station	DS
S105	C3	Pooja	Sadashiv Peth	Oracle

- Here no student has enrolled for the course Dot Net, hence no entry for course Dot Net.

B) Update Anomaly

- An **Update Anomaly** occurs when there is requirement of changes in multiple records of an entity, but all records not get updated.
- For Example : Address of Kunal get changed.

Table 2-Q. 3.14 : STUDENT_INFO

STUD_ID	COURSE_ID	STU_NAME	ADDRESS	COURSE
S101	C1	Kunal	Tilak Road	VB
S101	C2	Kunal	Camp	Java
S102	C3	Radhika	Sahkar Nagar	Oracle
S103	C4	Kiran	Tilak Road	C++
S104	C5	Jay	Pune Station	DS
S105	C3	Pooja	Sadashiv Peth	Oracle

Q. 3.15 What is normalization ? Enlist types of it. What is the need of normalization ? **(5 Marks)**

Ans. :

- **Normalization** is a database design technique which is used to organize the tables in such manner that it should reduce redundancy and dependency of data.

Types of Normalization

(1) **First normal form (1NF) :** Having unique values, no repeating groups.

(2) **Second Normal form (2NF) :** Having unique values, no repeating groups, no partial dependency.

(3) **Third Normal form (3NF) :** Same like second normal form and having transitive dependency.

(4) **Boyce-Codd Normal form (BCNF) :** It is more developed version then 3NF.

(5) **Fourth normal form (4NF) :** No multi-valued dependency.

Need of Normalization

- In database management process without Normalization, it is not easy to manage database operations like insertion, deletion, and modification without facing data loss problem.

- If database is not normalized, then Insertion, Updation and Deletion Anomalies occur frequently.

Q. 3.16 One of the rule designed by codd's for good relational database management system is integrity independence, which states that all integrity constraints can be independently modified without the need of any change in the application. Justify the significance of rule in relational database management system. **(5 Marks)**

Ans. :

- The integrity constraints must be specified independently from application programs and stored in the catalog.
- We should be able to make changes in integrity constraints independently without the need of any change in the application.
- ***Integrity independence*** is the ability to change integrity constraints without changing update transactions and application programs (Codd, 1990).
- Except for entity and referential integrity rules, notions of the relational data model are not rich enough to represent other types of constraints into database schemata.
- Thus in relational DBMSs integrity independence is either missing or restricted.
- We provide this capability to relational databases on an abstract level and not on the implementation level as discussed in (Codd, 1990).
- This means that integrity independence is provided using notions and operations of the relational data model and not depending on capabilities of DBMSs which may differ from one system to another.
- We do that by storing templates for simplified forms of constraints together with additional information about the constraints into meta relations.
- Meta relations are developed by adapting the method proposed by Nicolas in (Nicolas, 1982) for simplifying constraints in such a way that we can obtain the simplified constraints at compile time.
- The collections of meta relations define a meta database. The data model of the meta database is the relational data model.
- The main feature of the meta database is that depending on the updating operations of a given transaction that transforms a valid state into a new one, we can obtain from the meta database simplified forms for constraints that could be violated by these operations.
- This is achieved by using a specific relational algebra expression and a general substitution.

Q. 3.17 Write a short note on elements of the design process. **(5 Marks)**

Ans. :

The basic elements of the design process are :

1. Defining the problem or objective
2. Researching the current database
3. Designing the data structures
4. Constructing database relationships
5. Implementing rules and constraints
6. Creating database views and reports
7. Implementing the design

1. Defining the problem or objective

- This is the first and important step of database design in which we will address the problem or objective of the database. Here the nature of data to be stored is decided.

2. Researching the current database

- In some cases, there may be some database already in exist. Such database may be in any format like written on papers, spreadsheets, word files etc.
- The existing database is always useful for the end user. This existing database helps to determine the essential data structure of the database.

3. Designing the data structures

- A database system is always a set of data tables, hence the next step in the design process is to identify and describe those data structures.
- The tables in the database represents some distinct subject or any physical object. By determining the subjects and objects in the system we can create list of tables to design. Then attributes of each table are decided.

4. Constructing database relationships

- After creation of data structures, we have to establish relationships between the databases. It is necessary that each table should have a unique key which can identify the individual records in that table.

❑❑❑

Key Notes

(Exam Point of View)

Exam Oriented Key Points

Chapter-1: Introduction to DBMS

Drawbacks of Traditional File Processing Systems

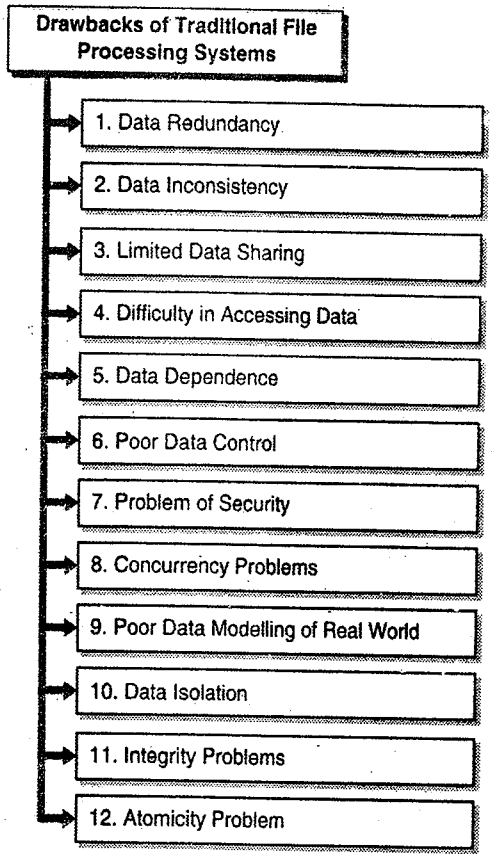


Fig. C1.1

Advantages of Database Management Systems

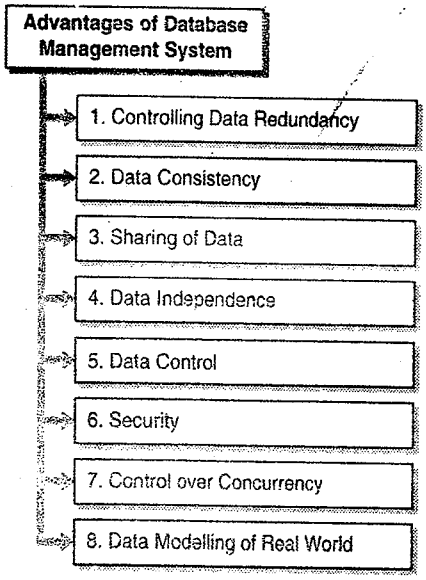


Fig. C1.2

Disadvantages of Database Management System

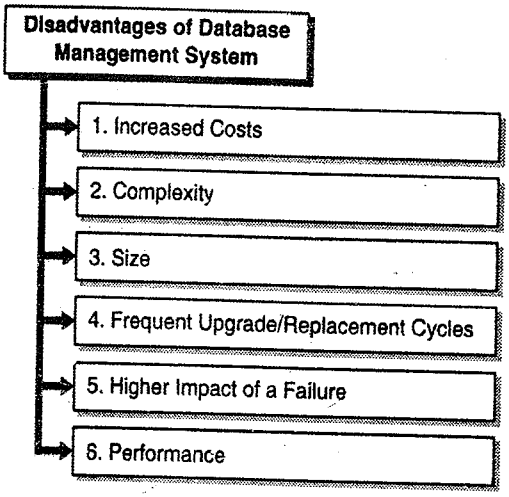


Fig. C1.3

Database-System

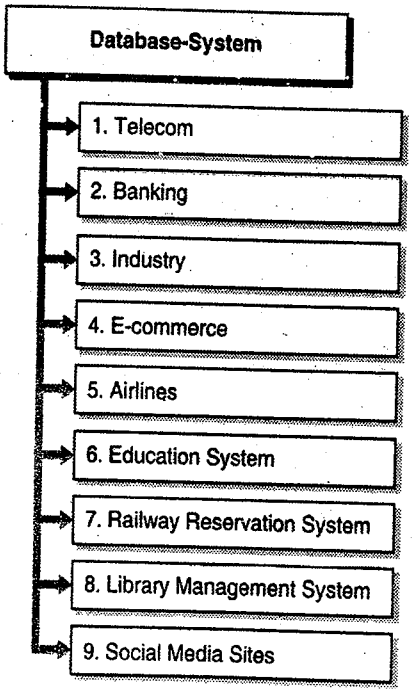


Fig. C1.4

Levels of Abstraction

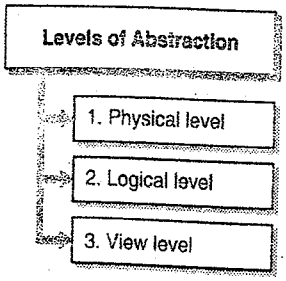


Fig. C1.5

☞ **Types of Schema**

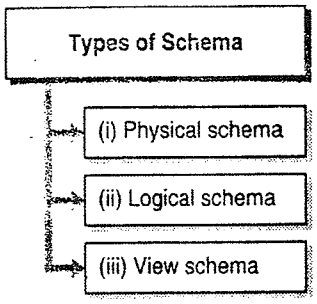


Fig. C1.6

☞ **Types of DML**

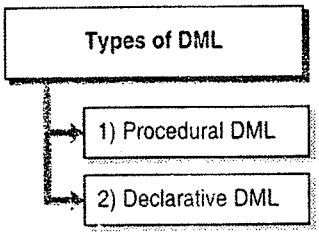


Fig. C1.7

☞ **Types of Data Models**

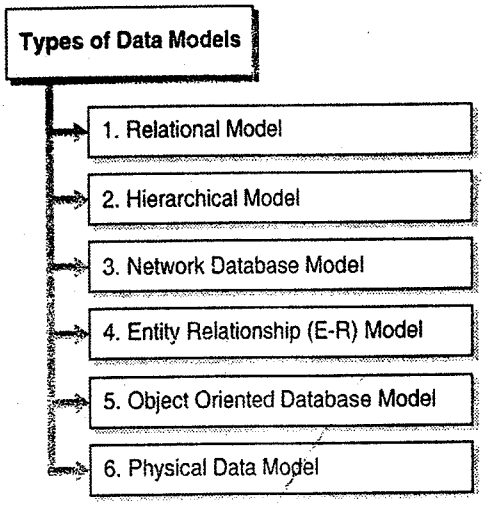


Fig. C1.8

☞ **Advantages of Relational Data Model**

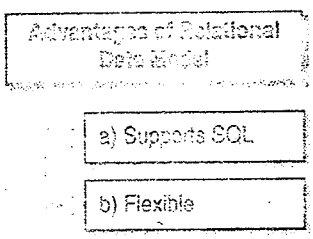


Fig. C1.9

☞ **Advantages of Hierarchical Model**

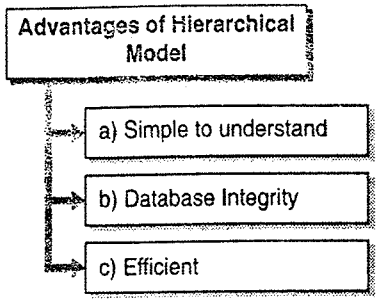


Fig. C1.10

☞ **Advantages of Network Model**

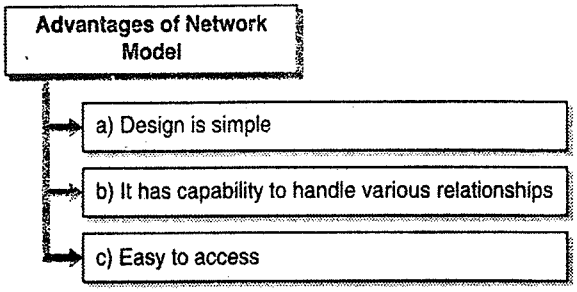


Fig. C1.11

☞ **Advantages of Entity Relationship Model**

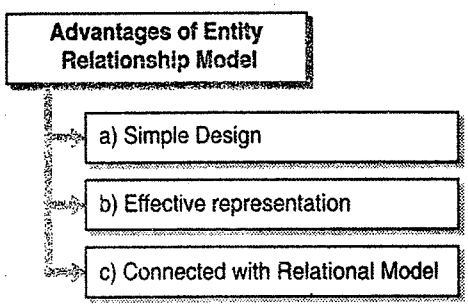


Fig. C1.12

☞ **Major problems avoid in database design schema**

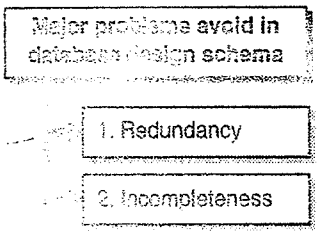


Fig. C1.13

☞ **Types of entities in Database management system**

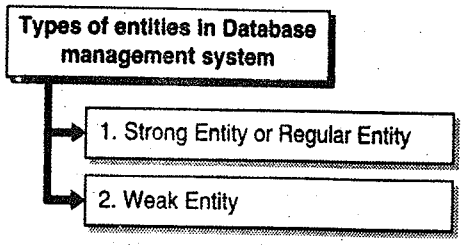


Fig. C1.14

☞ **Types of attributes in Database Management System**

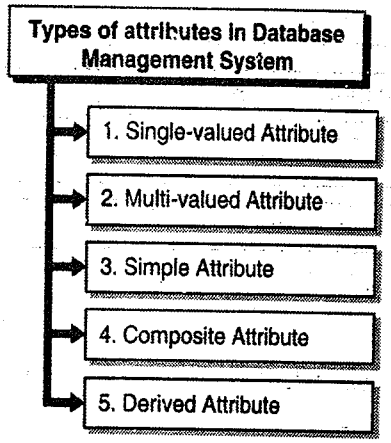


Fig. C1.15

☞ **The Degree of Relationships**

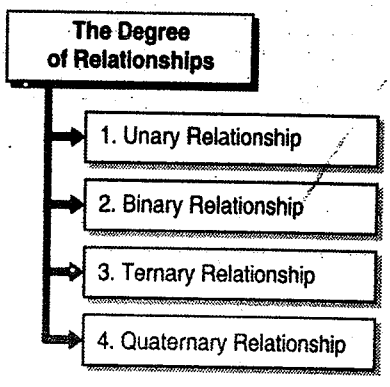


Fig. C1.16

☞ **Types of constraints**

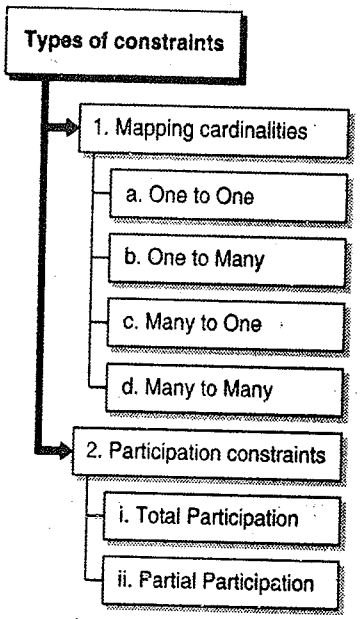


Fig. C1.17

☞ **Types of keys in DBMS**

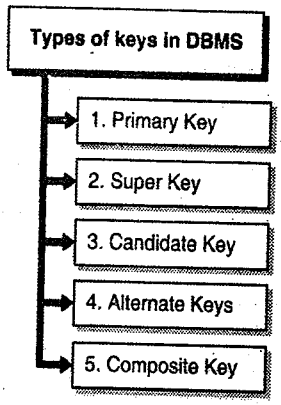


Fig. C1.18

☞ **Database Design Process**

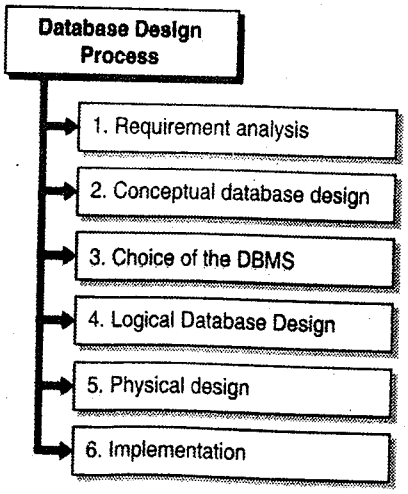


Fig. C1.19

Mapping Cardinality in E-R Diagram

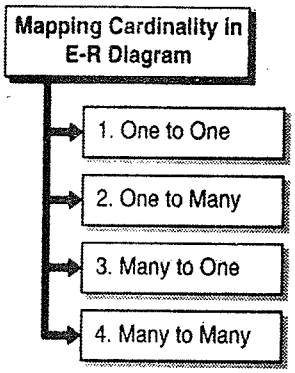


Fig. C1.20

Extended ER-Features

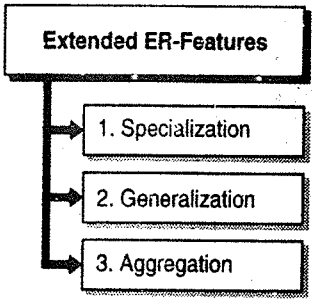


Fig. C1.21

Chapter 2 : SQL and PL/SQL

Advantages of SQL

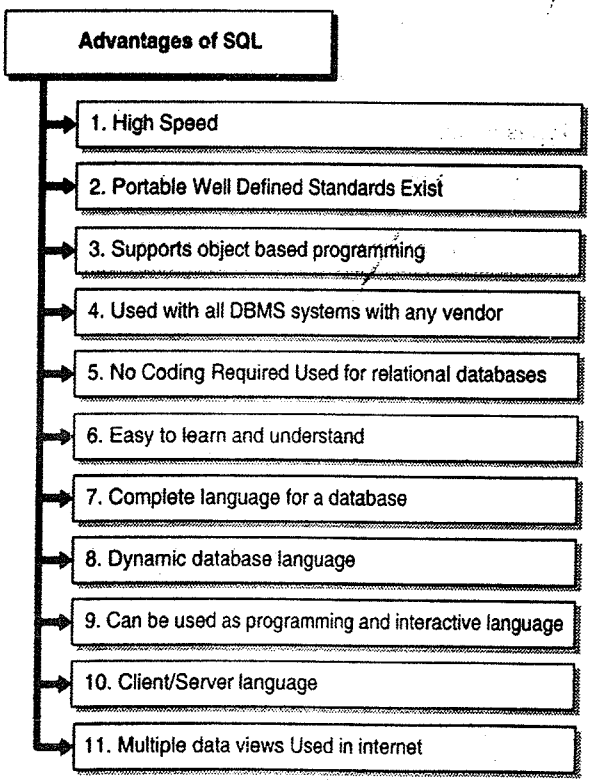


Fig. C2.1

Types of SQL Literals

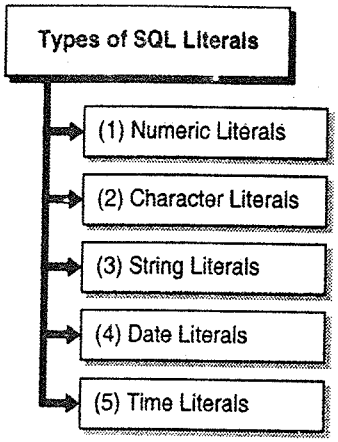


Fig. C2.2

Database Languages

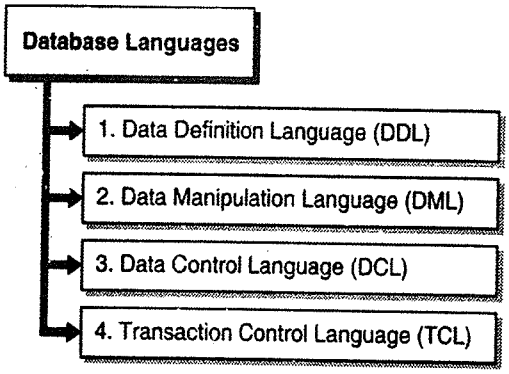


Fig. C2.3

SQL Operators

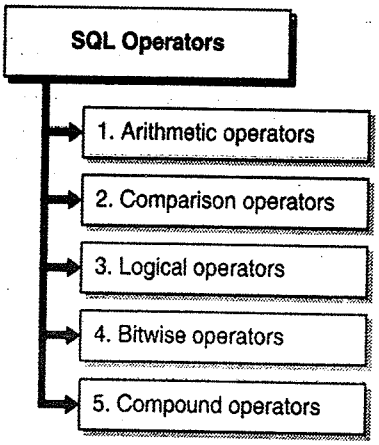


Fig. C2.4

Types of Indexes

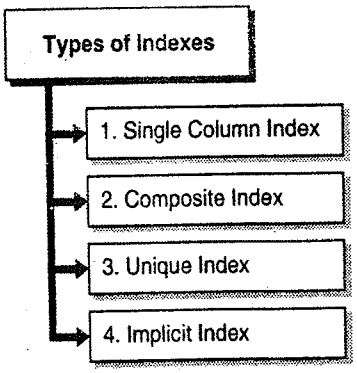


Fig. C2.5

Set Operations

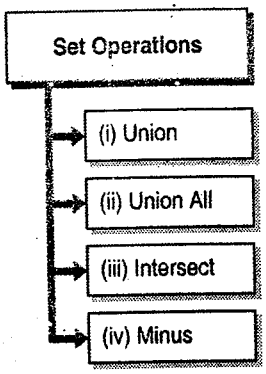


Fig. C2.6

Types of predicates

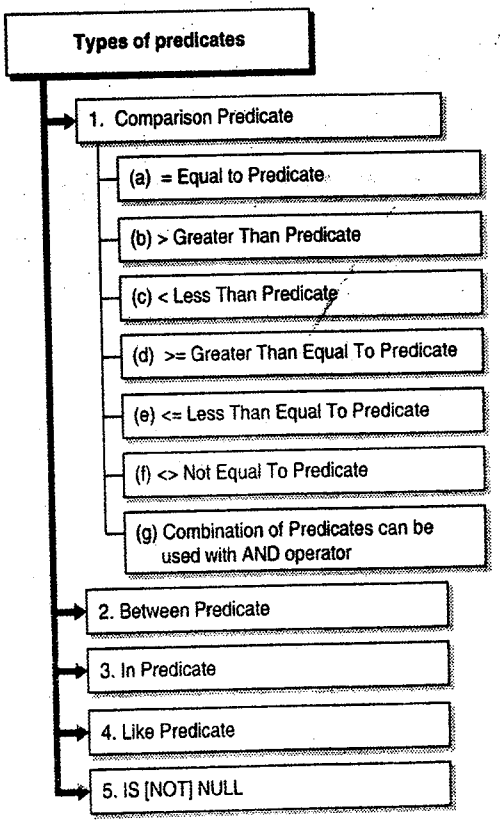


Fig. C2.7

Types of Joins

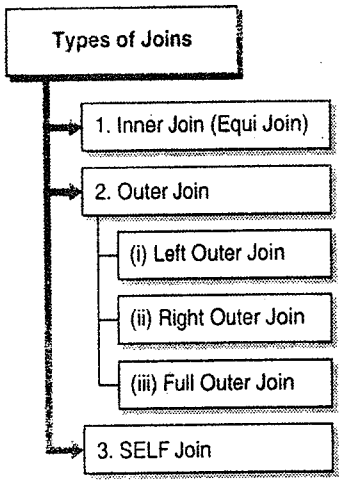


Fig. C2.8

Types of aggregate functions

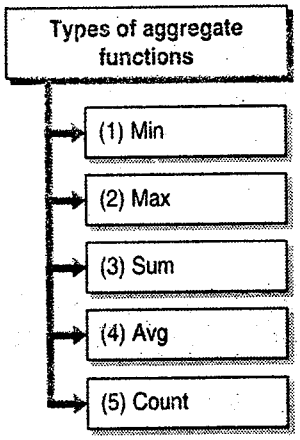


Fig. C2.9

Types of cursors

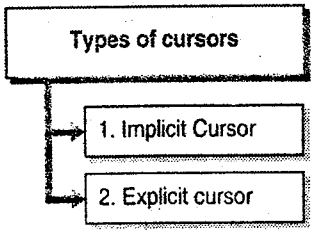


Fig. C2.10

☞ **Some concepts in Embedded SQL**

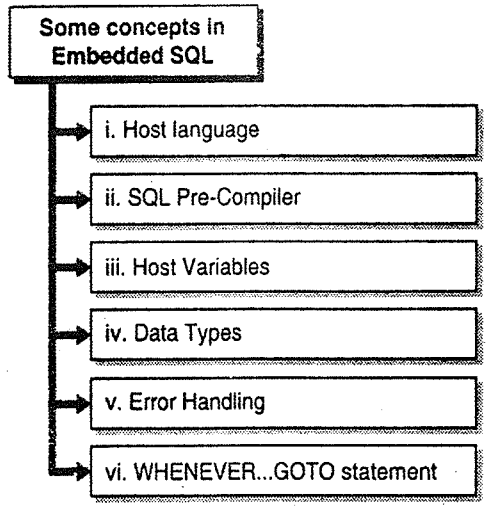


Fig. C2.11

Chapter 3 : Relational Database Design

☞ **Advantages of Relational Model**

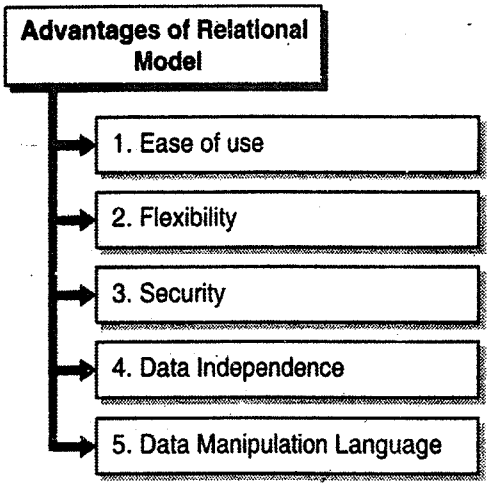


Fig. C3.1

☞ **Types of constraints**

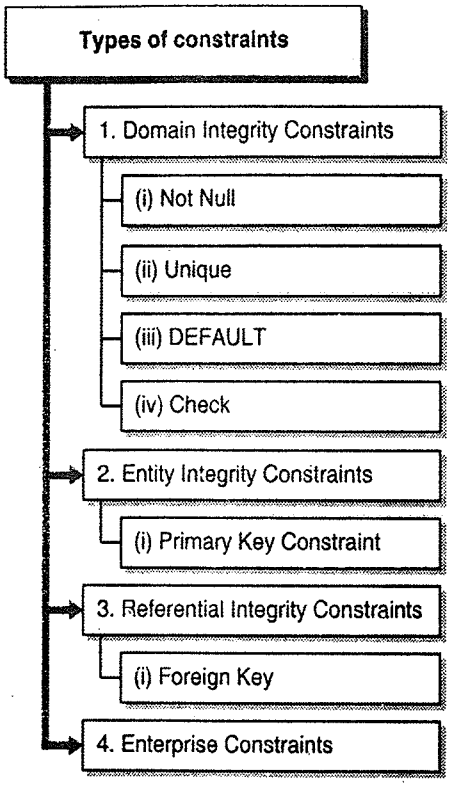


Fig. C3.2

☞ **Basic elements of the design process**

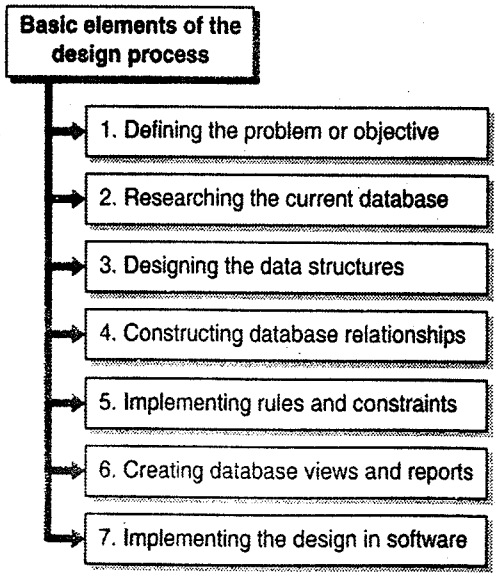


Fig. C3.3

☞ **Types of anomalies**

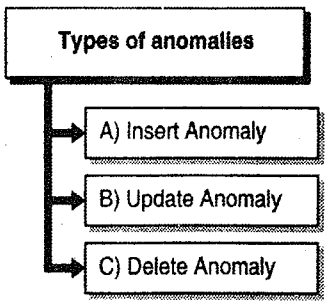


Fig. C3.4

☞ **Types of Functional Dependencies**

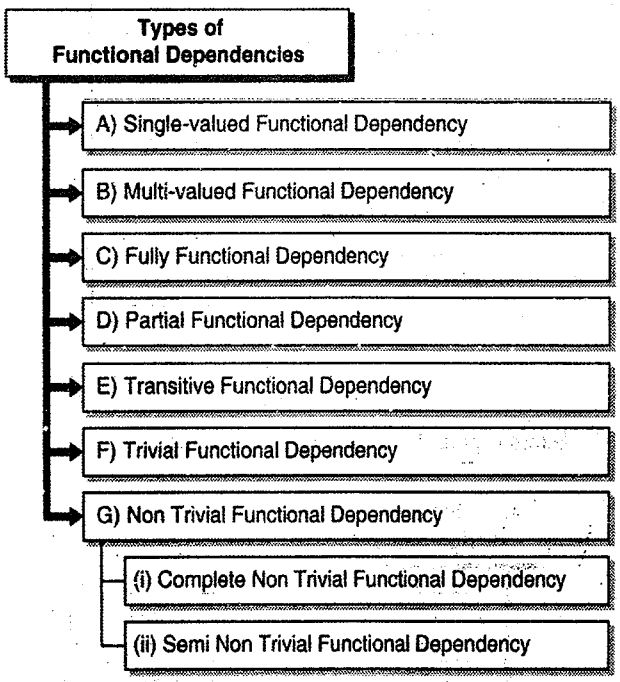


Fig. C3.5

☞ **Inference rules**

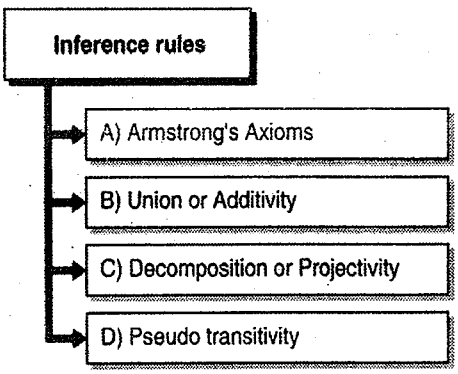


Fig. C3.6

☞ **Properties of decomposition**

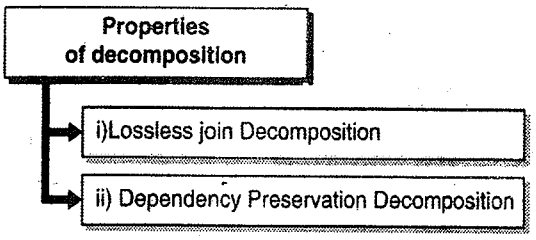


Fig. C3.7

Chapter 4 : Database Transactions and Query Processing

☞ **Process of Transaction**

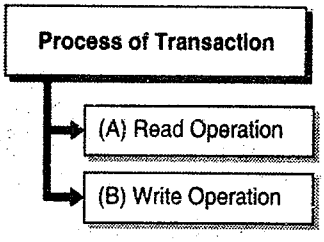


Fig. C4.1

☞ **ACID properties**

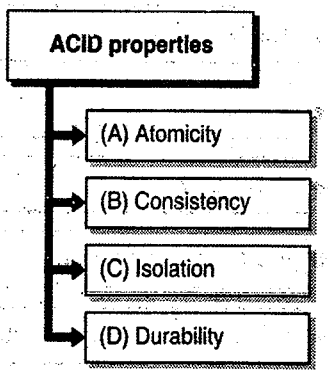


Fig. C4.2

☞ **Types of Schedule**

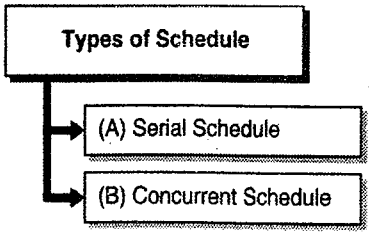


Fig. C4.3

☞ **Types of Serializability**

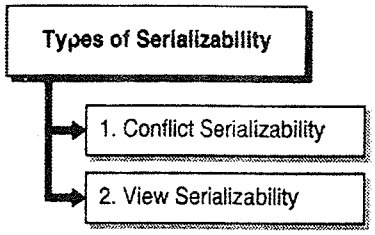


Fig. C4.4

☞ **Types of Schedules Based on Recovery**

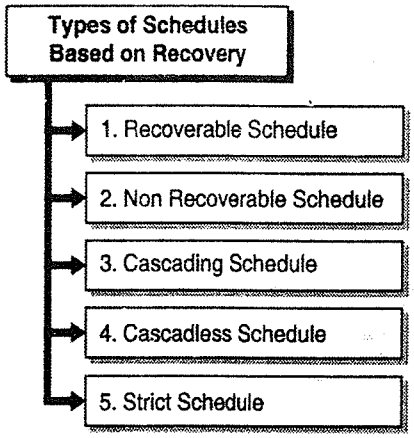


Fig. C4.5

☞ **Types of locks**

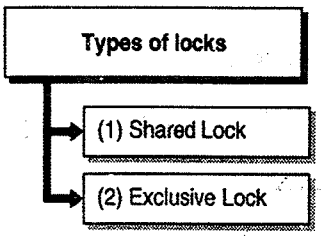


Fig. C4.6

☞ **Lock Based Protocols**

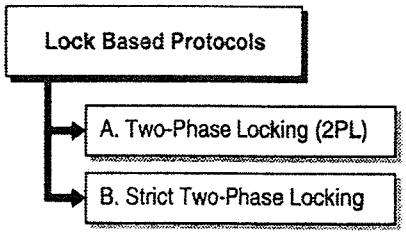


Fig. C4.7

☞ **Methods to handle deadlocks**

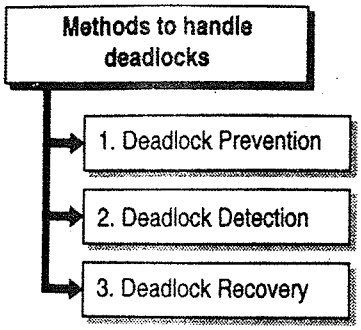


Fig. C4.8

☞ **Actions in Deadlock Recovery**

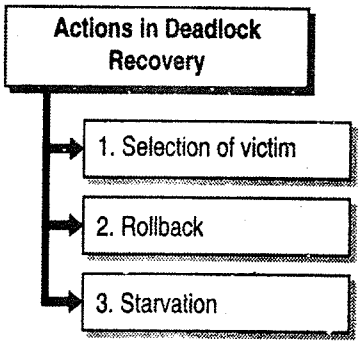


Fig. C4.9

☞ **Recovery Methods**

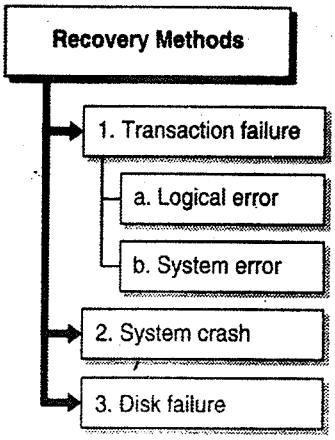


Fig. C4.10

☞ **Techniques to maintain the atomicity of transaction**

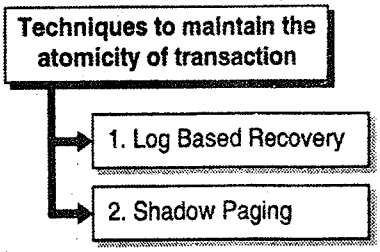


Fig. C4.11

☞ **Steps of query processing**

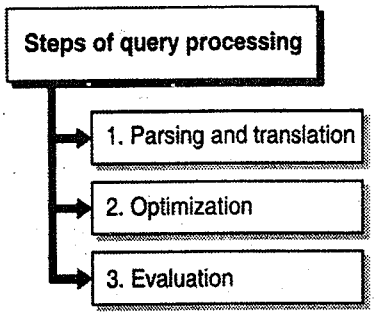


Fig. C4.12

Chapter 5 : Parallel and Distributed Databases

☞ **Common architectures to implement multi-user database management systems**

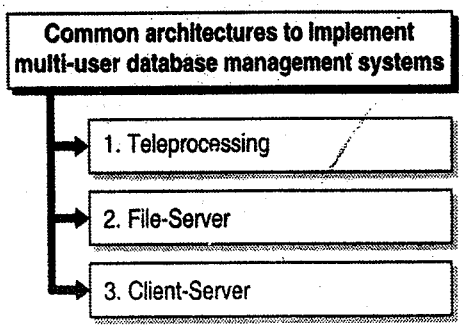


Fig. C5.1

☞ **Advantages of Client Server Architecture**

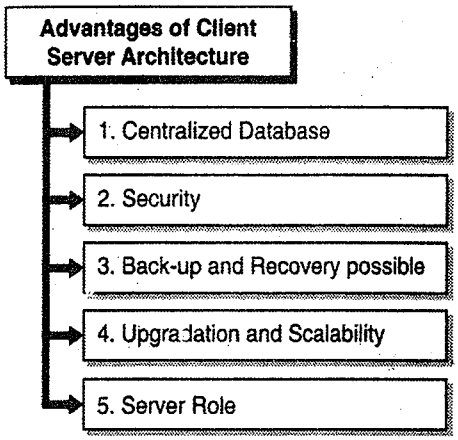


Fig. C5.2

☞ **Layers or Services**

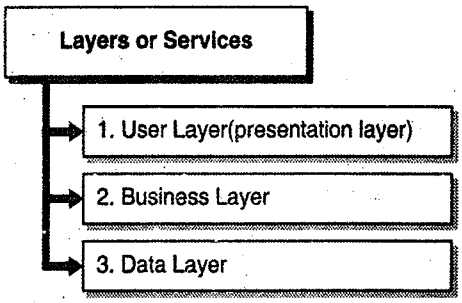


Fig. C5.3

☞ **Types of Database Architecture**

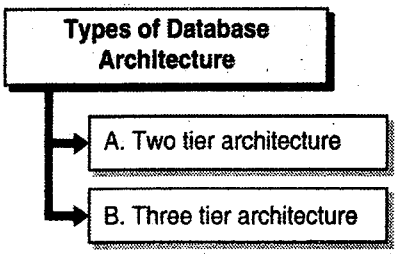


Fig. C5.4

☞ **Components of Oracle database**

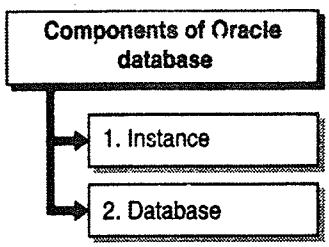


Fig. C5.5

☞ **Types of Database Structure**

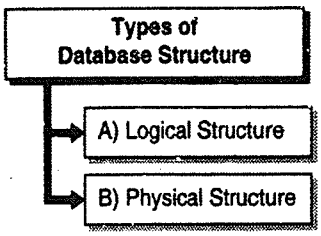


Fig. C5.6

☞ **Advantages of Parallel Database**

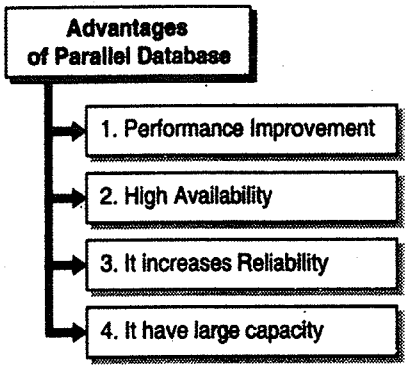


Fig. C5.7

☞ **Architectural models of parallel database**

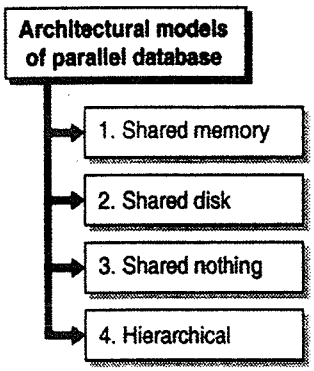


Fig. C5.8

☞ **Advantages of Distributed Database**

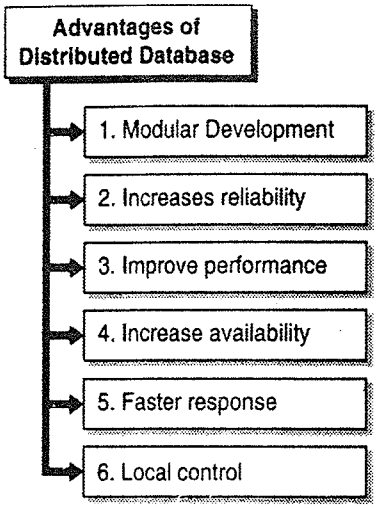


Fig. C5.9

☞ **Disadvantages of Distributed Database**

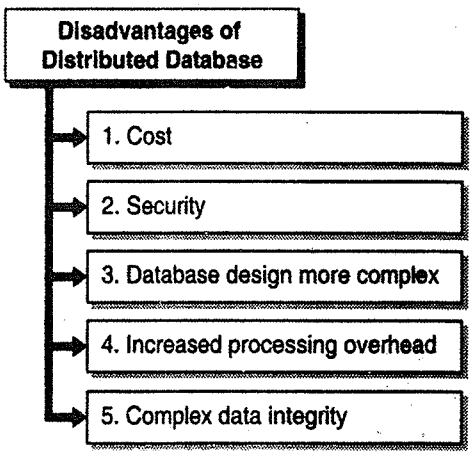


Fig. C5.10

☞ **Types of Distributed Database System**

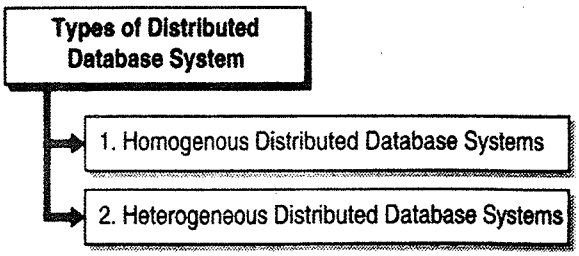


Fig. C5.11

☞ **Architectural Models**

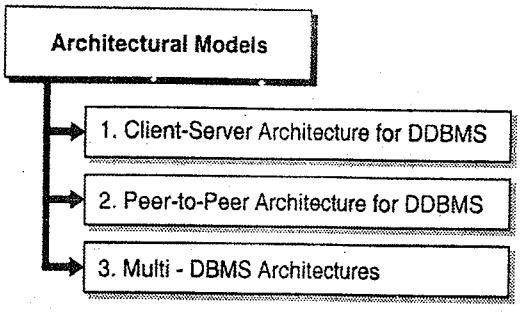


Fig. C5.12

☞ **Levels of schemas in architecture**

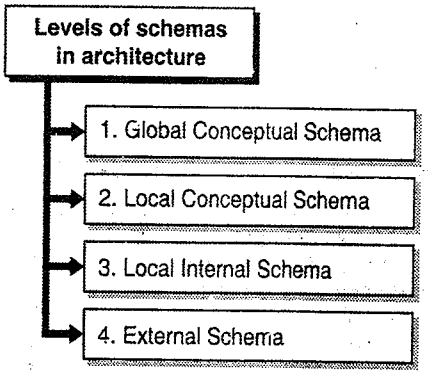


Fig. C5.13

☞ **Levels of schemas in Multi-DBMS**

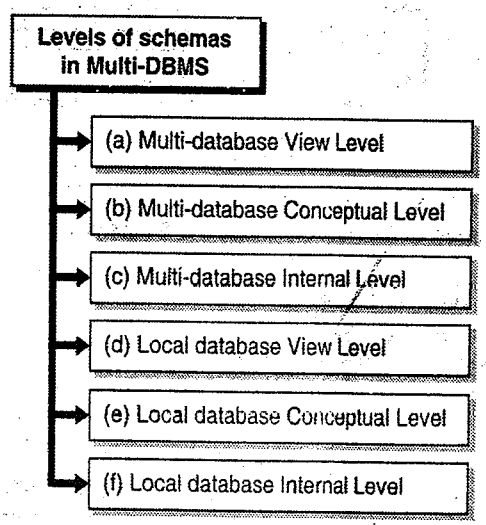


Fig. C5.14

☞ **Distributed Database Design**

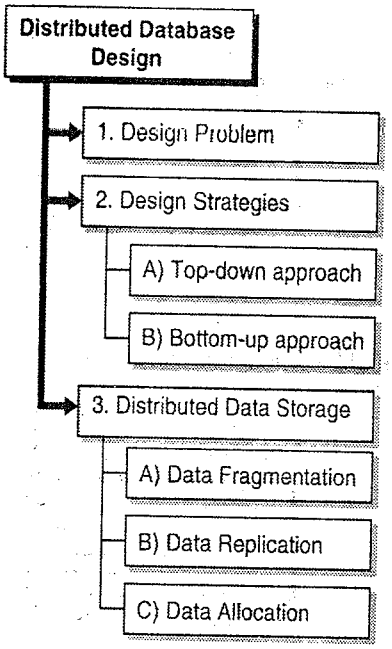


Fig. C5.15

☞ **Types of transaction**

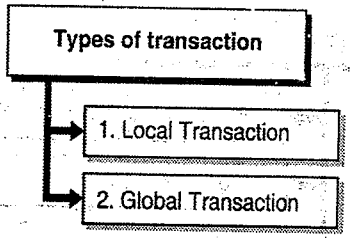


Fig. C5.16

☞ **Distributed Transactions Failure Modes**

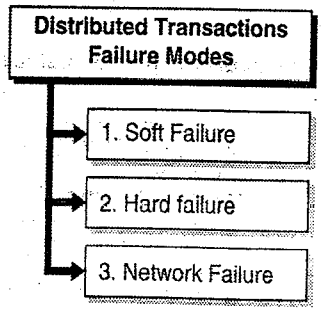


Fig. C5.17

☞ Commit protocols

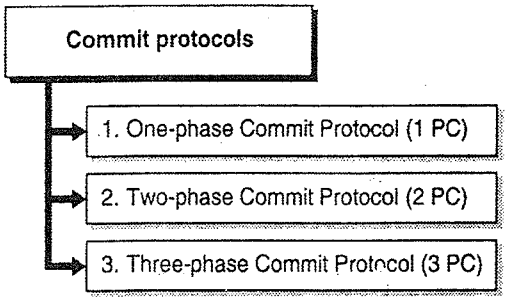


Fig. C5.18

☞ Approaches for concurrency control in distributed database

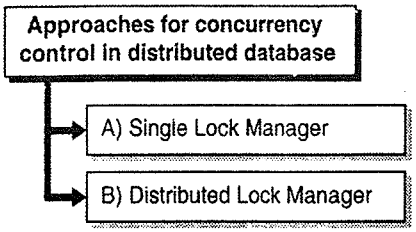


Fig. C5.19

Chapter 6 : NoSQL Database

☞ Advantages of NoSQL

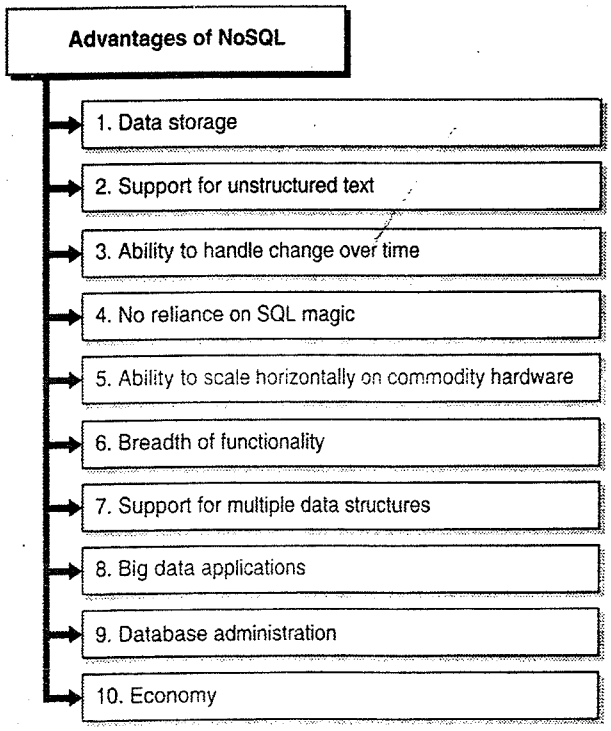


Fig. C6.1

☞ Types of NoSQL

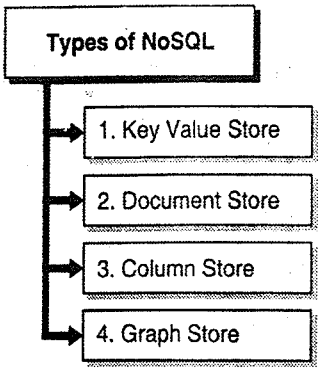


Fig. C6.2

☞ BASE Properties

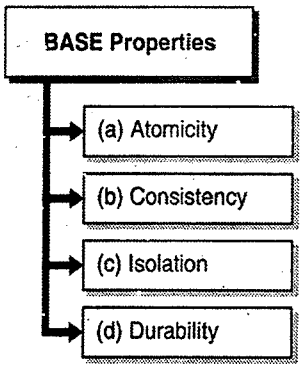


Fig. C6.3

☞ Data category

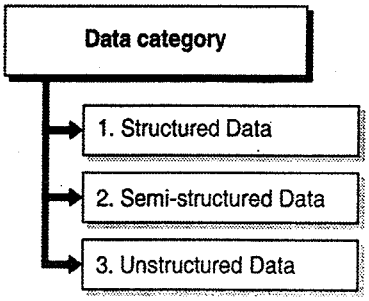


Fig. C6.4

☞ **Issues regarding data in traditional file storage system**

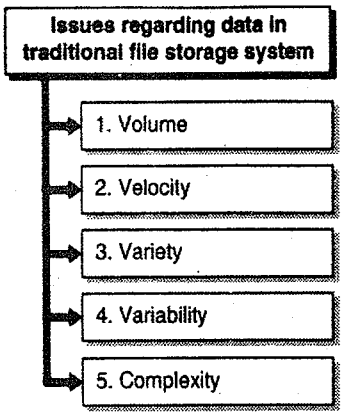


Fig. C6.5

☞ **Modules (Components) of Hadoop**

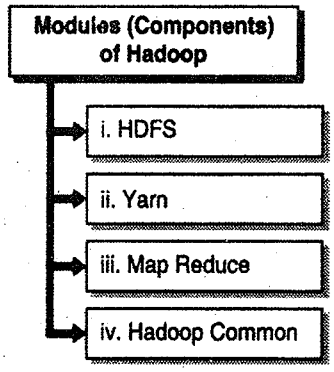


Fig. C6.6

❑❑❑

Fully Solved University Question Papers

- Aug. 2017 (In Sem)
- Dec. 2017

August 2017

Q. 1 A University registrar's office maintains data about the following entities :

- courses, including number, title, credits, syllable, and prerequisites;
- course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
- students, including student_id, name, and program; and
- instructors, including identification number, name, department and title;

Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

(i) Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.

(ii) Construct appropriate tables for E-R diagram designed with above requirements. **(Chap 1, 10 Marks)**

Ans. : [Hint : Please refer Example as Ex. 1.11.7]

(i)

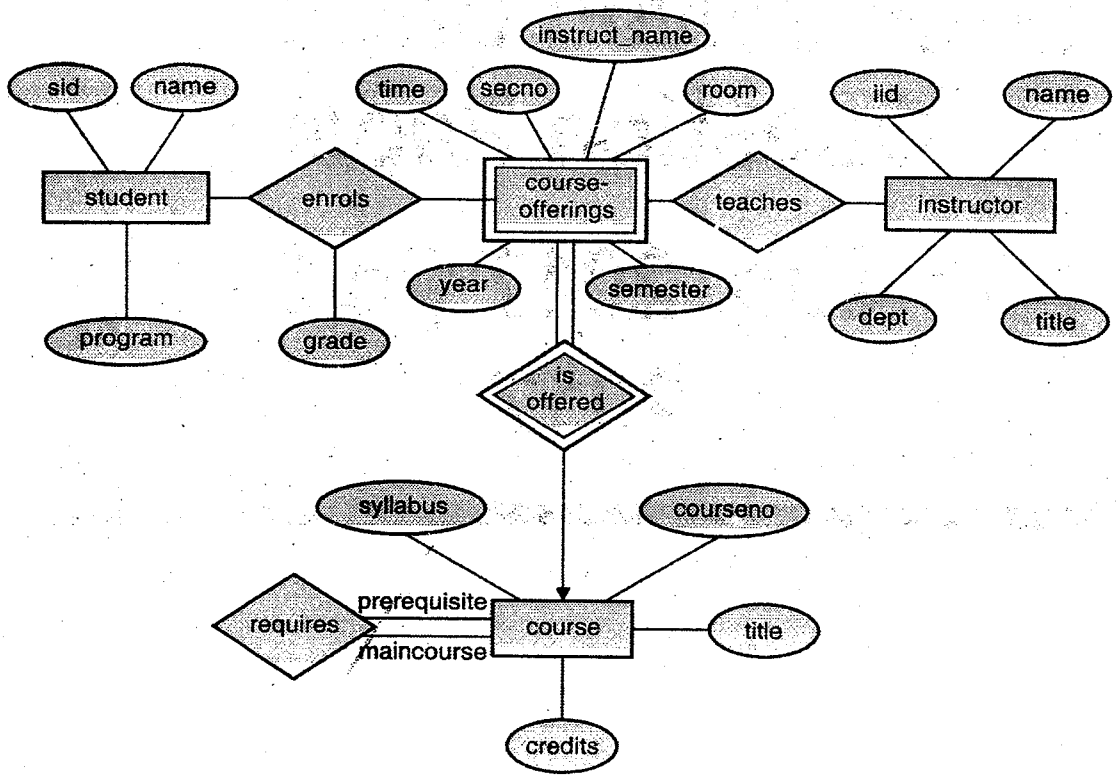


Fig. 1-Q. 1 : E-R diagram for university

(ii) Student(sid, name, program)

Course (Courseno, title, credits, syllable, prerequisite)

Instructor(iid, name, dept ,title)

Course offerings (courseno, year, semester, secno, time, room, instruct_name)

OR

Q. 2(a) A weak entity set can always be made into strong entity set by adding to its attributes; the primary key attributes of its identifying entity set. Outline what sort of redundancy will result if we do so while converting into tables.
(Ans. : Refer section 1.11)
(Chap 1, 5 Marks)

Ans. :

The primary key of a weak entity set can be inferred from its relationship with the strong entity set. If we add primary key attributes to the weak entity set, they will be present in both the entity set and the relationship set and they have to be the same. Hence there will be redundancy.

Consider the following example

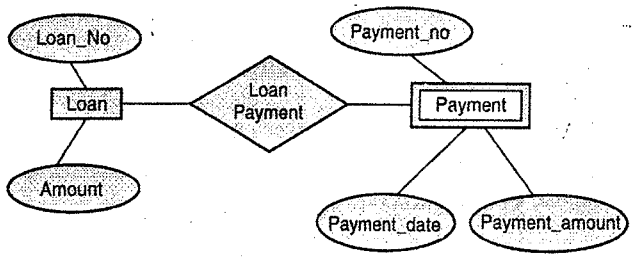


Fig. 1-Q. 2(a) : Weak Entity Set

Consider following tables

Loan

Loan_No	Amount
L101	35000
L105	50000
L107	70000
L115	90000

Payment

Payment_no	Loan_No	Payment_Amount	Payment_date
P111	L101	7000	11-04-2017
P101	L105	10000	2-08-2017
P211	L107	7000	21-12-2017
P313	L115	9000	25-05-2017

- Here payment is weak entity set to make it strong entity set we add composite primary key as (Payment_no, Loan_No).
- Here Loan_No will be present in both the entity set and the relationship set and they have to be the same. Hence there will be redundancy.

Q. 2(b) Explain with example what is physical data independence. Also explain its importance.
(Ans. : Refer section 1.4.1.1) **(Chap 1, 5 Marks)**

Ans. :

Physical data independence

- Logical level is the next higher level of abstraction which is used to describe what data the database stores, and what relationships exist in between the data items.
- The logical level thus describes an entire database in terms of a small number of relatively simple structures.
- Although implementation of the simple structures at the logical level may involve complex physical level structures, the user of the logical level does not need to be aware of this complexity. This is considered as physical data independence.
- The ability to change the physical schema without changing the logical schema is called physical data independence.
- **For example,** a change to the internal schema, such as using different file organization or storage structures, storage devices, or indexing strategy, should be possible without having to change the conceptual or external schemas.
  - Physical data independence points out the physical storing patterns changes by undamaging conceptual structures or arrangements. The presence of internal level in the architecture of database and the operation of changes from the conceptual level to internal level achieves the physical data independence.
  - Mapping between conceptual level and internal level provides a way to propagate from conceptual records to physical or stored records. If sophistication is made in the physical devices then likewise changes should

be made in mapping of conceptual level and internal level which maintains conceptual level unchanged. To make conceptual schema as physically independent of data then external patterns defined on conceptual schema should be physical data independent

Importance of Physical data Independence

- Ability of improving performance
- Alterations in data structure does not requires alterations in application programs
- Implementation details can be hidden from the users
- Reduction of incongruity
- Tractability in improvement of system
- Affordable prices of maintaining system
- Providing the best services to the users
- Permit users to focus on general structure
- Enforcement of standards
- Improvement of security
- The state of being undamaged or undivided can be improved.

Q. 3(a) Consider the following database schema:

Physician(reg_no,name,tel_no,city)

Patient(p_name,street,city)

Visit(p_name,reg_no,date_of_visit,fee)

Write SQL queries for following requirements (any 2)

(i) Find the name and city of patients who visited a physician on 13 July 2017.

(ii) Get the name of physician and the total no. of patients visited him.

(iii) Get the details of date wise fees collected at clinic. **(Chap 2, 5 Marks)**

Ans. :

(i) Select p.p_name, p.city from patient p, visit v where p.p_name=v.p_name.

And v.date_of_visit =" 13 July 2017";

(ii) Select distinct ph.name, count(p.name) from visit v, physician ph patient p where ph.reg_no=v.reg_no and p.p_name=v.p_name.

(iii) Select ph.reg_no, ph.name, p.p_name, v.fee, v.date_of visit from visit v, patient p, physician ph where ph.reg_no=v.reg_no and p.p_name=v.p_name.

Q. 3(b) Write short note on Embedded SQL along with its applications. *(Ans. : Refer section 2.21)* **(Chap 2, 5 Marks)**

Ans. :

Embedded SQL

- The method of combining of general purpose programming languages and database language like SQL is called as **embedded SQL.**
- The SQL is good for defining database structure and defining short queries but for some application it is required to mix SQL with programming language.
- In other words, SQL defines WHAT is required but it can't define HOW to meet this requirement.
- SQL is used to express queries but all the queries can't be expressed with SQL alone, so there is a need of combining database language with general purpose programming language to express such queries.

Some concepts in Embedded SQL

- **Host language :** These are programming languages (such as C, C++, COBOL, Java, etc) in which SQL statements are embedded.
- **SQL Pre-Compiler :** A pre-compiler is used to process embedded SQL statements. It is used to translate SQL statements into DBMS library calls which can be implemented into host language.
- The following code is a simple embedded SQL program, written in C

The program accepts order number from user, retrieves the customer number, salesperson, and status of the order, and displays the data on the screen.

```
int main()
{
  EXEC SQL INCLUDE SQLCA;
  EXEC SQL BEGIN DECLARE SECTION;
    int Order_ID;
    int Cust_ID;
    char Sales_Person[10]
```

```
    char Order_Status[6]
  EXEC SQL END DECLARE SECTION;
  /* Set up error processing */
  EXEC SQL WHENEVER SQLERROR GOTO
qry_error;
  EXEC SQL WHENEVER NOT FOUND GOTO
invalid_number;
  /* Prompt the user for order number */
  printf ("Enter order number: ");
  scanf_s("%d", &Order_ID);

  /* Execute the SQL query */
  EXEC SQL SELECT Cust_ID, Sales_Person,
Order_status
    FROM Orders
    WHERE Order_ID = :Order_ID
    INTO :Cust_ID, :Sales_Person, :Order_Status;
  /* Display the results */
  printf ("Customer number: %d\n", Cust_ID);
  printf ("Salesperson: %s\n", Sales_Person);
  printf ("Status: %s\n", Order_Status);
  exit();

qry_error:
  printf ("SQL error: %ld\n", sqlca->sqlcode);
  exit();

invalid_number:
  printf ("Invalid order number.\n");
  exit();
}
```

- **Host Variables :** These variables are declared in section starting with BEGIN DECLARE SECTION and ending with END DECLARE SECTION keywords. These variables are prefix by (:) in an embedded SQL statement. The precompiler distinguish between host variables and database objects(like column and tables) using (:).
- **Data Types :** Usually the data types of DBMS and a host language are different. This affects the host variable because they are used by both by host language statements and embedded SQL statements. When there is no similar type in host language that corresponds to a DBMS data type, the DBMS automatically converts the data.
- **Error Handling :** Run times errors are reported by DBMS to the application program using SQL Communications Area, or SQLCA. In the above program the statement INCLUDE SQLCA includes the SQLCA structure in program. It is used when errors are processed by the programs which are returned by the DBMS.
- **WHENEVER...GOTO statement :** It tells the pre-compiler to generate error-handling code when an error occurs.

Applications of Embedded SQL

1. In embedded SQL, SQL is used to handle the database and the host language handles the variables and other input and output functions.
2. In the development cycle tasks like parsing and optimizing are done which take high overheads. It increases efficiency of CPU resources.
3. Portability is achieved. The application be precompiled on one machine and can be moved another machine for further processing.
4. There is no need that programmer should understand the complex structure of DBMS. He has created only the embedded SQL source program code.

OR

Q. 4(a) What is index created on table column ? How performance of SELECT query is improved if index is created on table ?
(Ans. : Refer section 2.7) **(Chap 2, 5 Marks)**

Ans. :

- Sometimes the data in the database is very large. For example in the application of State Bank of India, the database related to customers and their transactions is very large. In such case retrieval of data from such huge database becomes slower.
- Indexes are the special lookup tables which are available to only database search engine for accessing data. Indexes speed up data retrieval effectively.
- An index is a pointer to data in a table. It is similar to the alphabetical index of a book present at the end of book. An index is used to speed up SELECT queries and also WHERE clauses.

- But because of indexes the data input related to INSERT and UPDATE statements get slow down.
- Indexes can be created or dropped with no effect on the data.

Creating Index

1. Single Column Index

This is index is created on a single column of a table.

Syntax

```
CREATE INDEX index_name
ON table_name (column_name)
```

For Example

```
CREATE INDEX ind1
on student(stud_name);
```

2. Composite Index

Sometimes duplicate records may available in columns. In such case the composite indexing is better option to index the data. This index is created on a multiple columns of a table.

Syntax

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

For Example

```
CREATE INDEX ind2 on student(stud_name ,
marks);
```

3. Unique Index

A unique index does not allow any duplicate values to be inserted into the table.

Syntax

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

For Example

```
CREATE UNIQUE INDEX ind3
on student(stud_name);
```

4. Implicit Index

Implicit indexes are indexes that are automatically created by the database server when an object is created. Such indexes are created for primary key and unique constraints.

Displaying Index

To display index information regarding table following query is used.

Syntax

```
Show index from table_name;
```

For Example

```
Show index from student;
```

- The indexes are special objects which built on top of tables. The indexes can do an operation like SELECT, DELETE and UPDATE statement faster to manipulate a large amount of data. An INDEX can also be called a table and it has a data structure. An INDEX is created on columns of a table.
- INDEXES can locate information within a database very fast.
- The INDEX first sorts the data and then it assigns an identification for each row.
- The INDEX table having only two columns, one is a rowid and another is indexed-column (ordered).
- When data is retrieved from a database table based on the indexed column, the index pointer searches the rowid and quickly locates that position in the actual table and display, the rows sought for.

Q. 4(b) Write PL/SQL block of code which accepts the roll no. from user, the attendance of roll no entered by user will be checked in stud_att(Roll_no,Att) table. Attendance of Roll no entered is displayed on Screen.

(Chap 2, 5 Marks)

Ans. :

```
DECLARE
xstud_rollno number(5);
xstud_att number(5);
xtotal_days number(3):=200;
BEGIN
```

```
xstud_rollno:=&xstud_rollno;
select Att into xstud_att from stud_att where
Roll_no=xstud_rollno;
dbms_output.put_line('Att is '||xstud_att );
END IF;
END;
```

Q. 5(a) Explain what is normalization? Explain with example requirements of Third Normal Form. *(Ans. : Refer sections 3.5 and 3.9)*

(Chap 3, 5 Marks)

Ans. :

Normalization

- Dr. Edgar F. Codd proposes normalization as an integral part of a relational model. **Normalization** is a database design technique which is used to organize the tables in such manner that it should reduce redundancy and dependency of data.
- **Normalization** is a database design technique which is used to organize the tables in such manner that it should reduce redundancy and dependency of data.
- It divides larger tables to smaller tables and links these smaller tables using their relationships.
- Normalization is implemented by following some formal rules either by a process of synthesis or decomposition. Synthesis is used to create database design in normalized form based on a known set of dependencies. Decomposition generally done on existing database which is not normalized. The relation of this database is further divided into multiple relations to improve the design.
- Normalization is the multi step process. It creates smaller tables from larger table.

Third Normal Form (3NF)

A database design is said to be in **3NF** if both the following conditions are satisfied by it

- Initially the design must be in 2NF.
- No non-prime attribute should be transitively dependent on prime key attribute.
- For any **non-trivial** functional dependency, $X \rightarrow A$

  Either X is a superkey or A is prime attribute.

  Consider the Table 1-Q. 5(a).

Table 1-Q. 5(a) : STUDENT_DETAILS

STUD_ID	STU_NAME	CITY	ZIP
S101	Kunal	Pune	411037
S102	Radhika	Nasik	422001
S103	Kiran	Mumbai	400016
S104	Jay	Nagpur	440001

- In Table 1-Q. 5(a) the STUD_ID is the only one primary key. Here the data of city can be retrieved through either STUD_ID or ZIP code. But the CITY is not superkey as well as nor the CITY is prime attribute.
- Here the CITY is depends upon ZIP and ZIP is depends upon STUD_ID
- $\text{Stud\_id} \rightarrow \text{zip} \rightarrow \text{city}$.
- This relationship is known as **transitive dependency**.
- We have to remove this transitive dependency by implementing Third Normal Form. For this purpose we have to split the STUDENT_DETAILS table in two different tables say STUDENT_DATA and CITY.

STUDENT_DATA

STUD_ID	STU_NAME	CITY
S101	Kunal	Banking System
S102	Radhika	Library Management
S103	Kiran	Speech to Text Converter
S104	Jay	ATM

CITY

ZIP	CITY
411037	Pune
422001	Nasik
400016	Mumbai
440001	Nagpur

- Now the database design is converted into Third Normal Form. Here the basically design is already in Second Normal Form and No non-prime attribute is transitively dependent on prime key attribute.

Q. 5(b) Explain with example the concept of referential integrity constraint (e.g. Foreign key in SQL). Also discuss the situations when referential integrity constraint is getting violated by Insert, Update and Delete operations on table. *(Ans. : Refer section 3.3.3)* **(Chap 3, 5 Marks)**

Ans. :

Referential Integrity Constraint

- The Foreign key constraint is also known as Referential Integrity Constraint. In this constraint one field is common in between two tables.
- Foreign key represent relationships between tables. There is parent child relationship between two tables having common column.
- The master table can be referenced as parent while the transaction table is considered as child. The common field will work as primary key in parent table while foreign key in child table.

Example

Consider Training Institute Database having two tables Course_details and Student. There is a condition that the students may register for courses which are available in institute currently and not for the courses which are not offered at the moment. To specify this rule while inserting values into database foreign key constrain is used. As follows :

Course_details (Master / Parent) Table

Course_Code	Course_Name	Fees
Or	Oracle	5000
Jv	Java	4000
Cp	C Programming	3000

Student(Transaction / child) Table

Stud_ID	Name	Course_Code
S101	Kunal	Jv
S102	Radhika	Or
S013	Kiran	Cp

- In both the tables, the field **Course_Code** is common. In Course details Course_Code is referred as primary key and in Student table it is referred as foreign key.
- So, after assigning foreign key constraint to Student table the record entry for new student will not accept Course_Code which is not available in the master table Course_details.

Rules

When an SQL operation attempts to change data in such a way that referential integrity will be compromised, a referential constraint could be violated. For example,

- An insert operation could attempt to add a row of data to a child table that has a value in its foreign key columns that does not match a value in the corresponding parent table's parent key.
- An update operation could attempt to change the value in a child table's foreign key columns to a value that has no matching value in the corresponding parent table's parent key.
- An update operation could attempt to change the value in a parent table's parent key to a value that does not have a matching value in a child table's foreign key columns.
- A delete operation could attempt to remove a record from a parent table that has a matching value in a child table's foreign key columns.
- The database manager handles these types of situations by enforcing a set of rules that are associated with each referential constraint. This set of rules consists of :
  1. An insert rule
  2. An update rule
  3. A delete rule

1. Insert rule

- The insert rule of a referential constraint is that a non-null insert value of the foreign key must match some value of the parent key of the parent table. The value of a composite foreign key is null if any component of the value is null.
- This rule is implicit when a foreign key is specified.

2. Update rule

- The update rule of a referential constraint is specified when the referential constraint is defined.

The choices are NO ACTION and RESTRICT. The update rule applies when a row of the parent or a row of the dependent table is updated.

- In the case of a parent row, when a value in a column of the parent key is updated, the following rules apply :
- If any row in the dependent table matches the original value of the key, the update is rejected when the update rule is RESTRICT.
- If any row in the dependent table does not have a corresponding parent key when the update statement is completed the update is rejected when the update rule is NO ACTION.

3. Delete rule

- The delete rule of a referential constraint is specified when the referential constraint is defined. The choices are NO ACTION, RESTRICT, CASCADE, or SET NULL. SET NULL can be specified only if some column of the foreign key allows null values.
- If the identified table or the base table of the identified view is a parent, the rows selected for delete must not have any dependents in a relationship with a delete rule of RESTRICT, and the DELETE must not cascade to descendent rows that have dependents in a relationship with a delete rule of RESTRICT.
- If the delete operation is not prevented by a RESTRICT delete rule, the selected rows are deleted. Any rows that are dependents of the selected rows are also affected:
- The nullable columns of the foreign keys of any rows that are their dependents in a relationship with a delete rule of SET NULL are set to the null value.
- Any rows that are their dependents in a relationship with a delete rule of CASCADE are also deleted, and the above rules apply, in turn, to those rows.

OR

Q. 6(a) Explain different features of good relational database design.
(Ans. : Refer section 3.4.1) **(Chap 3, 5 Marks)**

Ans. : Features of Good Relational Designs

- It should be able to distribute the information into different tables to reduce redundancy of data.
- It should provide easy access to the data available in multiple tables.
- It should take care of accuracy and integrity of information.
- It should provide functionalities for data processing and reporting needs.
- Data entry, updates and deletions should be efficient.
- Data retrieval, summarization and reporting should also be efficient.
- The database design must be self-documenting since much of the information is stored in the database rather than in the application.
- To understand the features of a good relational design we will see an example.

Example

- The relational design of Banking
  - Branch:(brnch_name,brnch_city,assets)
  - customer : (cust_id, cust_name, cust_street, cust_city)
  - loan : (loan_num, amt)
  - account : (acc_num, balance)
  - employee : (emp_id, emp_name, telephone_num, start_date)
  - dependent_name : (emp_id, dname)
  - account_branch : (acc_num, brnch_name)
  - loan_branch : (loan_num, brnch_name)
  - borrower : (cust_id, loan_num)
  - depositor : (cust_id, acc_num)
  - cust_banker : (cust_id, emp_id, type)
  - works_for : (worker_emp_id, manager_emp_id)
  - payment : (loan_num, payment_num, payment_date, payment_amt)
  - savings_account : (acc_num, interest_rate)
  - checking_account : (acc_num, overdraft_amount)
- Consider we want to get information from borrower and loan, then after combining these two relations -

  bor_loan = (cust_id, loan_num, amt)

- In the result, repetition of information is possible.
- Result is possible repetition of information. Observe **Ln101** in Fig. 1-Q. 6(a).

(A) Combine Schemas

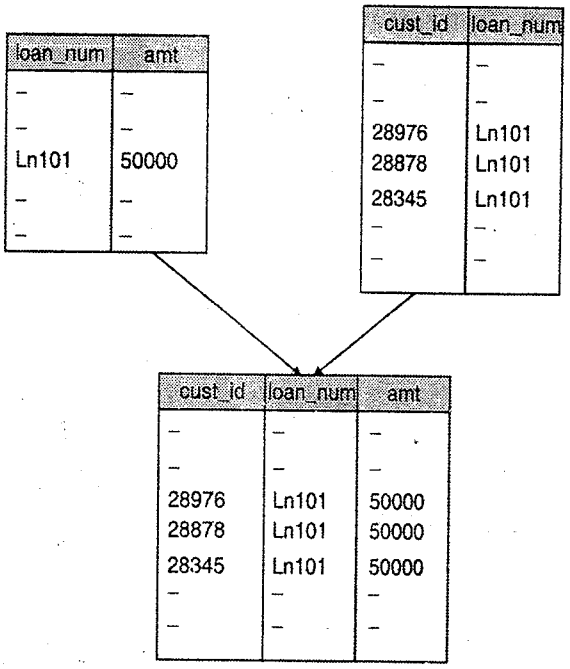


Fig. 1-Q. 6(a)

1. Combining schema without repetition

- Consider we want to get information from loan_branch and loan, then after combining these two relations -

loan_amt_br = (loan_num, amt, brnch_name)

- Here no repetition of data will occur.

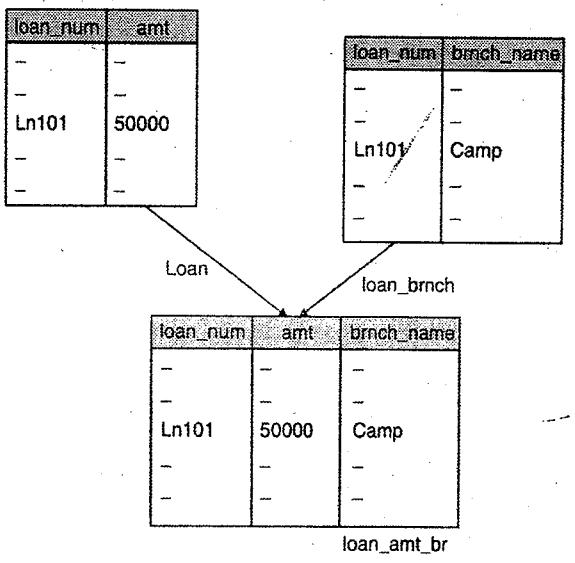


Fig. 2-Q. 6(a)

B) Smaller Schemas

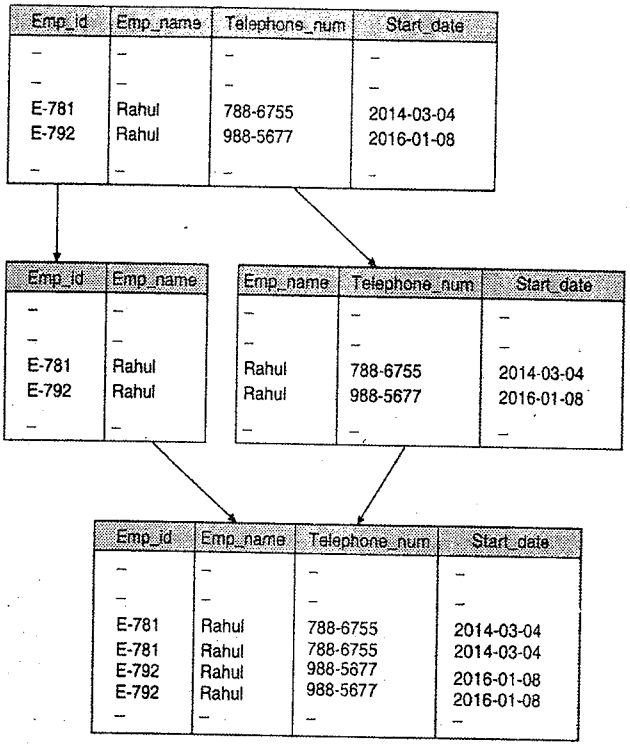


Fig. 3-Q. 6(a)

- Consider that we have created bor_loan first. Now we want to split (decompose) it in borrower and loan.
- Decide a rule for a schema (loan_num, amt), loan_num would be a candidate key.
- Denote as a functional dependency : amount→loan_number
- In bor_loan relation the loan_num is not a candidate key hence the amount of a loan may have to be repeated. This indicates that we have to decompose bor_loan.
  - It is not necessary that always all decompositions are good.
  - Suppose we decompose employee into

employee1 = (emp_id, emp_name)

employee2 = (emp_name, telephone_num, start_date)

See the Fig. 3-Q. 6(a) which indicates how we lose information. We cannot reconstruct the original employee relation. This is called as lossy decomposition.

Q. 6(b) One of the rule designed by codd's for good relational database management system is integrity independence, which states that all integrity constraints can be independently modified without the need of any change in the application. Justify the significance of rule in relational database management system.
(Ans. : Refer section 3.2) **(Chap 3, 5 Marks)**

Ans. :

- The integrity constraints must be specified independently from application programs and stored in the catalog.
- We should be able to make changes in integrity constraints independently without the need of any change in the application.
- **Integrity independence** is the ability to change integrity constraints without changing update transactions and application programs (Codd, 1990). Except for entity and referential integrity rules, notions of the relational data model are not rich enough to represent other types of constraints into database schemata.
- Thus in relational DBMSs integrity independence is either missing or restricted. We provide this capability to relational databases on an abstract level and not on the implementation level as discussed in (Codd, 1990).
- This means that integrity independence is provided using notions and operations of the relational data model and not depending on capabilities of DBMSs which may differ from one system to another.
- It implemented by storing templates for simplified forms of constraints together with additional information about the constraints into meta relations.
- Meta relations are developed by adapting the method proposed by Nicolas in (Nicolas, 1982) for simplifying constraints in such a way that It can obtain the simplified constraints at compile time.
- The collections of meta relations define a meta database. The data model of the meta database is the relational data model.
- The main feature of the meta database is that depending on the updating operations of a given transaction that transforms a valid state into a new one, it can obtain from the meta database simplified forms for constraints that could be violated by these operations.
- This is achieved by using a specific relational algebra expression and a general substitution.

Dec. 2017

Q. 1(a) Consider a database used to record the marks that students get in different exams of different course offerings. Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database.

(Chap 1, 5 Marks)

Ans. :

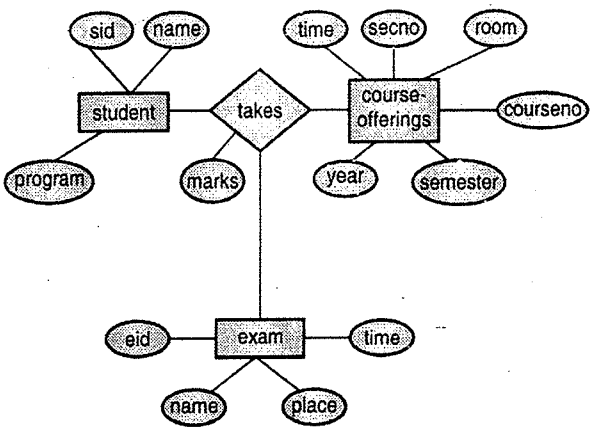


Fig. 1-Q. 1(a)

Q. 1(b) For the database system to be usable, it must retrieve data efficiently. The need of efficiency has led designers to use complex data structures to represent data in the database. Developers hides this complexity from the database system users through several levels of abstraction. Explain those levels of abstraction in detail.
(Ans. : Refer section 1.4.1) **(Chap 1, 5 Marks)**

Ans. :

Abstraction

- Abstraction is an important feature of Database Management System. Extracting the important data by ignoring the remaining irrelevant details is known as **abstraction**.
- Database systems are usually made-up of complex data structures as per their requirements.

- To make the user interaction easy with database, the internal irrelevant details can be hidden from users. This process of hiding irrelevant details from user is called data abstraction.
- The complexity of database can be hiding from user by different level of abstraction as follows.

Levels of Abstraction

There are three levels of abstraction :

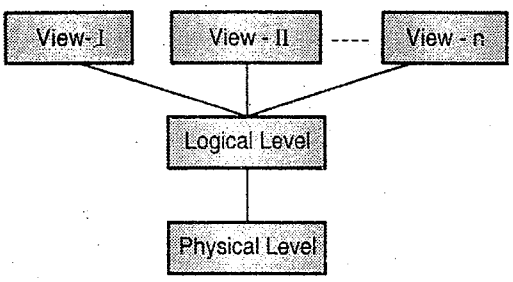


Fig. 1-Q. 1(b)

1. Physical level

- In data abstraction Physical level is the lowest level. This level describes how the data is actually stored in the physical memory.
- The physical memory may be hard disks, magnetic tapes, etc. In Physical level the methods like hashing are used for organization purpose.
- Developer would know the requirement, size and accessing frequency of the records clearly in this level which makes easy to design this level.

2. Logical level

- This is the next higher level of abstraction which is used to describe what data the database stores and what relationships exist in between the data items.
- The logical level thus describes an entire database in terms of a small number of relatively simple structures.
- Although implementation of the simple structures at the logical level may involve complex physical level structures, the user of the logical level does not need to be aware of this complexity.
- This is considered as physical data independence.
- Database administrators use the logical level of abstraction to decide what information to keep in a database.

3. View level

- It is the highest level of data abstraction. This level describes the user interaction with database system. In the logical level, simple structures are used but still complexity remains because in the large database various type of information is stored.
- Many users are not aware of technical details of the system, and also they need not to access whole information from the database.
- Hence it is necessary to provide a simple and short interface for such users as per their requirements. Multiple views can be created for same database for multiple users.
- **Example** : Consider that we are storing information of all the employees of an organization in employee table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are usually hidden from the developer.
- The records can be described as fields and attributes along with their data types at the **logical level**. The relationship between these fields can be implemented logically. Usually the developer works at this level because they have knowledge of such things about database systems.
- End user interacts with system with the help of GUI and enters the details at the screen at **view level**. User is not aware of how the data is stored and what data is stored; such details are hidden from them.

OR

Q. 2(a) Construct an alternative E-R diagram for above requirements given in Q. 1(a) that uses only a binary relationship between students and course-offerings. Make sure that only one relationship exists between a particular student and course-offering pair, yet you can represent the marks that a student gets in different exams of a course offering.

(Chap 1, 5 Marks)

Ans. :

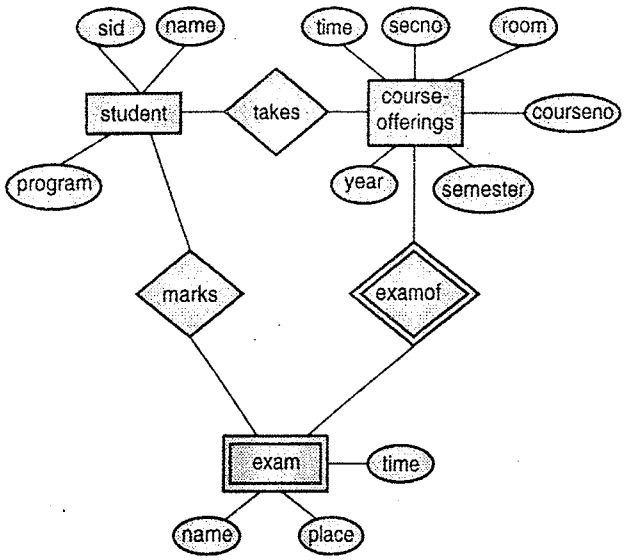


Fig. 1-Q. 2(a)

Q. 2(b) Write PL/SQL trigger for following requirement:

Event : Deletion of row from stud (roll_no, name, class) table.

Action : after deletion of values from stud table, values should be inserted into cancel_admission

(roll_ no, name) table,

Note : for every row to be deleted, action should be performed. **(Chap 2, 5 Marks)**

Ans. :

```
CREATE or REPLACE TRIGGER trigger4
  AFTER DELETE ON STUDENT
  FOR EACH ROW
BEGIN
Insert into cancel_admission values(:OLD.ID,
:OLD.NAME, :OLD.CLASS);
END;
```

Output

```
the deleted row from student is inserted in
cancel_admission.
```

Q. 3(a) Consider insurance database with following schema :

person(driver-id, name, address)

car(license, model, year)

accident(report - no, date, location)

owns(driver-id,license)

participated(driver-id, car, report-no, damage-amount)

Write a query in SQL for following requirements.

i) Find the total no. of people who owned cars that were involved in accidents in 2016.

ii) Retrieve the name of person whose address contains Pune.

iii) Find the name of persons having more than two cars. **(Chap 2, 5 Marks)**

Ans. :

i) Select count (distinct name)

from accident, participated, person

where accident. report-no=participated. report-no

and participated. driver-id=person. driver-id

and date between date'2016-00-00'and date'2016-12-31'

ii) Select name from person

where address like '% pune%'

iii) Consider, person(driver-id, name, address, number_of_cars)

Select name from person

Where number_of_cars<2;

Q. 3(b) Any database system to be good relational database system, codds have proposed 12 rules, explain any 2 rules proposed by codd with example.

(Ans. : Refer section 3.2) **(Chap 3, 5 Marks)**

Ans. :

Codd designed these rules to define what is required from the relational database management system. All these thirteen rules are followed by very few commercial products. Even the oracle follows eight and half rules out of thirteen.

Codd's rule are

Rule 0 : Foundation Rule

Foundation rule states that the system must be capable to manage their database systems through their relational capabilities. All other twelve rules are derived from this foundation rule.

Remaining twelve rule are as follows :

Rule 1 : Information Representation

- All the information in the database must be stored in standard form of tables.
- Table is considered as best format to store and manage the data.

Rule 2 : Systematic treatment of null values

- In a database the NULL values must be given a systematic and uniform treatment. In number of cases we may have to set null values on place of data.
- For Example, data is missing, data is not known, or data is not applicable.
- Null Value is different from empty character or string, string of a blank character, and it is also different from zero value or any number.

Example

EMPLOYEE			
EMP_ID	NAME	DEPARTMENT	PH_NO
E101	Naren	Testing	9656865232
E102	Atharv	NULL	9421589654
E103	Sarang	Testing	NULL

In relation PH_NO of Sarang is NULL.

Rule 3 : The guaranteed access rule

- In a data base every single data element must be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value).
- There should not be requirement of any other entity to access the data.

Rule 4 : Active online catalog

- The description of the structure of entire database must be stored in an online catalog, known as data dictionary.
- Data dictionary can be accessed by authorized users. Metadata should be maintained for all the data in the database.
- The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language.
- That is nothing but, just like ordinary data the database description is represented at the logical level, so that is possible for authorized users to apply the same language of regular data to its interrogation.

Rule 5 : The comprehensive data sub language rule

Database should not be directly accessible. It should always be accessible by using some strong query language. This rule illustrates that the system should support at least one relational language.

The language -

a) Has a linear syntax

b) Can be used both interactively and within application programs,

c) 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).

Rule 6 : View updating rule

- Views are the virtual tables created using queries to show the partial or complete view of the table.

- All views that are theoretically updatable must also be updatable by the system
- The rule states that we should be able to make changes in views.

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

- High Level means multiple rows from multiple columns are affected by the single query.
- This rule states that a database must support insertion, updation, and deletion.
- This must not be limited for a single row, that is, it must also support union, minus and intersection operations to yield sets of data records.

For example,

- Suppose employees got 5% hike in a year. Then their salary has to be updated to reflect the new salary. Since this is the annual hike given to the employees, this increment is applicable for all the employees.
- Hence, the query should not be written for updating the salary one by one for thousands of employee. A single query should be strong enough to update the entire employee's salary at a time.

Rule 8 : Physical data independence

Changes in the physical level (i.e. format of data storage or container of data like array or linked list) must not require any change to an application.

Rule 9 : Logical data independence

- The user's view (application) should not be dependent upon logical data in a database.
- That means changes in logical data must not affect the applications which uses it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application.

Rule 10 : Integrity independence

- The integrity constraints must be specified independently from application programs and stored in the catalog.
- We should be able to make changes in integrity constraints independently without the need of any change in the application.

Rule 11 : Distribution independence

- The distribution of database at various locations should not be visible to end user.
- Users should always get the impression that the data is located at one site only.
- That means even though the data is distributed, it should not affect the speed of access and performance of data compared to centralized database.

Rule 12 : The non-subversion rule

- If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language.
- Means anyhow those integrity rules and constraints must be followed, we cannot violet them by using any back door option.

OR

Q. 4(a) What is normalization? What's the need of normalized database?
(Ans. : Refer section 3.5) **(Chap 3, 5 Marks)**

Ans. : Normalization

- Dr. Edgar F. Codd proposes normalization as an integral part of a relational model. **Normalization** is a database design technique which is used to organize the tables in such manner that it should reduce redundancy and dependency of data.
- **Normalization** is a database design technique which is used to organize the tables in such manner that it should reduce redundancy and dependency of data.
- It divides larger tables to smaller tables and links these smaller tables using their relationships.
- Normalization is implemented by following some formal rules either by a process of synthesis or decomposition.
- Synthesis is used to create database design in normalized form based on a known set of dependencies. Decomposition generally done on

existing database which is not normalized. The relation of this database is further divided into multiple relations to improve the design.

- Normalization is the multi step process. It creates smaller tables from larger table.

Need of Normalization

- In database management process without Normalization, it is not easy to manage database operations like insertion, deletion, and modification without facing data loss problem.
- If database is not normalized, then Insertion, Updation and Deletion Anomalies occur frequently. To understand need of normalization, first we should learn, what are **Anomalies?**

Anomalies

Anomalies are inconvenient or error-prone situation arising when we process the tables. There are three types of anomalies:

A) Insert Anomaly

An **Insert Anomaly** occurs when it is not possible to insert certain attributes into the database without the availability of other attributes. For example, In College Database System, it is not possible to add entry of any new course unless any student enrolled for it.

Table 1-Q. 4(a) : STUDENT_INFO

STUD_ID	COURSE_ID	STU_NAME	ADDRESS	COURSE
S101	C1	Kunal	Camp	VB
S101	C2	Kunal	Camp	Java
S102	C3	Radhika	Sahkar Nagar	Oracle
S103	C4	Kiran	Tilak Road	C++
S104	C5	Jay	Pune Station	DS
S105	C3	Pooja	Sadashiv Peth	Oracle

Here no student has enrolled for the course Dot Net, hence no entry for course Dot Net.

B) Update Anomaly

An **Update Anomaly** occurs when there is requirement of changes in multiple records of an entity, but all records not get updated.

For Example : Address of Kunal get changed.

Table 2-Q. 4(a) : STUDENT_INFO

STUD_ID	COURSE_ID	STU_NAME	ADDRESS	COURSE
S101	C1	Kunal	Tilak Road	VB
S101	C2	Kunal	Camp	Java
S102	C3	Radhika	Sahkar Nagar	Oracle
S103	C4	Kiran	Tilak Road	C++
S104	C5	Jay	Pune Station	DS
S105	C3	Pooja	Sadashiv Peth	Oracle

C) Delete Anomaly

A **Delete Anomaly** is opposite to insert anomaly. This anomaly occurs when data of some attributes get lost because of deletion of data of other attributes in the same record. For **Example :** Just consider the only one entry for course Oracle of student Jay is deleted, then there will be no record related to course 'DS'.

STUD_ID	COURSE_ID	STU_NAME	ADDRESS	COURSE
S101	C1	Kunal	Tilak Road	VB
S101	C2	Kunal	Camp	Java
S102	C3	Radhika	Sahkar Nagar	Oracle
S103	C4	Kiran	Tilak Road	C++
S105	C3	Pooja	Sadashiv Peth	Oracle

To overcome these anomalies we need to normalize the data in database.

Q. 4(b) The organization has decided to increase the salary of employees by 10% of existing salary, whose existing salary is less than Rs. 10000/- Write a PL/SQL block to update the salary as per above requirement, display an appropriate message based on the no. of rows affected by this update (using implicit cursor status variables). **(Chap 2, 5 Marks)**

Ans. :

```
SQL> begin
Update employee set salary = salary*0.10 where
salary<10000;
if sql%found then
dbms_output.put_line('Employee record modified
successfully');
else
```

```
dbms_output.put_line('Employee no. does not exist');
end if;
end;
```

Q. 5(a) Consider the Transaction (T3), Transaction (T4) and Transaction (T6) are any hypothetical transactions working on data item Q. Schedule explaining the execution of T3, T4 and T6 are given below. Decide whether following schedule is conflict Serializable or not ? Justify your answer.

T_3	T_4	T_6
read (Q)		
	Write (Q)	
Write (Q)		
		Write (Q)

(Chap 4, 9 Marks)

Ans. : Given schedule's :

T_3	T_4	T_6
read (Q)		
	Write (Q)	
Write (Q)		
		Write (Q)

Here Possible serial schedules for S is :

- $T_3 \rightarrow T_4 \rightarrow T_6$ (i.e. all the operations of T_3 will perform first then all the operations of T4 will perform and then all the operations of T_6 will perform),
- While transferring a schedule into another one following rule must be follow :
- If Oi and Oj are two operations in a transaction and Oi < Oj (Oi is executed before Oj), same order will follow in new schedule as well.
- **Swapping non-conflicting operations** $Write_1(Q)$ and $Write_2(Q)$ in S, the schedule becomes Schedule S_1.
- Schedule S1

T_3	T_4	T_6
Read1(Q)		
Write1(Q)		
	Write2(Q)	
		Write3(Q)

Similarly,

- **Swapping non-conflicting operations** W_1 (Q) and W_2 (Q) in S_1, the schedule becomes

Schedule $S1_2$

T_3	T_4	T_6
Read(Q)		
	Write(Q)	
Write(Q)		
		Write(Q)

- $S1_2$ is a serial schedule in which read operation of T_3 is performed then write operation of T_4 is performed, then again write operation of T_3 is performed, finally write operation on Q is performed on T_6.Since the S has been transformed into a serial schedule $S1_2$ by swapping non-conflicting operations of S. So, S is **conflict Serializable**.

Q. 5(b) Transaction during its execution should be in one of the different states at any point of time, explain the different states of transactions during its execution.
(Ans. : Refer section 4.3) **(Chap 4, 8 Marks)**

Ans. :

Transaction states

There are five states of transaction:

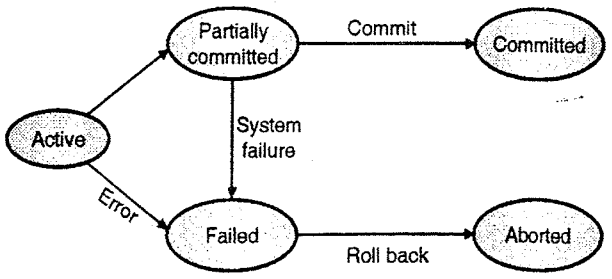


Fig. 1-Q. 5(b) : Transaction states

During execution, transaction will be in one of the following state.

Active

- This is the first state of transaction. In this state the transaction is being executed.
- This state is entry point to every transaction. Transactions which are in active state indicate that their execution has been started.

Partially Committed

When a transaction completes all operations, it will enter into partially committed state. Even the last operation has been performed; data is still not saved to the database.

Failed

If execution of transaction cannot proceed due to failure of the system or failure in database, then the transaction is said to be in failed state.

Abort

- In case of the failed transaction, the modification done in database during transaction processing must be rolled back to ensure the atomicity and consistency of database.
- The transaction enters into 'Abort' this state after roll back operation is performed. It is the end of transaction when any fault occurs.

Committed

- If without any error transaction completed successfully it will come into committed state which will allow to made changes permanent into database. This is the last step of a transaction, if it executes without fail.
- Every transaction goes through at least three states among of five. Either transaction may goes through active - partially committed - committed or through active – Failed - aborted, or through active - partially committed – failed - abort.

OR

Q. 6(a) Suppose a transaction T_i issues a read command on data item Q. How time-stamp based protocol decides whether to allow the operation to be executed or not using time stamp based protocol of concurrency control.
(Ans. : Refer section 4.8.2.3)
(Chap 4, 9 Marks)

Ans. : Timestamp Ordering Algorithm

- The timestamp-ordering protocol used to order the conflict read and write operation pairs in such a way that they maintain serializability. Timestamp ordering protocol ensures Serializability of conflict read/write operation pairs.
- This protocol ensures serializability by assuring that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.
  - The timestamp of transaction Ti is denoted as TS (Ti).
  - Read timestamp of data item A is denoted by R-timestamp (A).
  - Write timestamp of data item A is denoted by W-timestamp (A).

Basic Timestamp ordering protocol works as follows :

- Suppose that transaction Ti issue read(A) then either one of two conditions
  - If TS (Ti) < W-timestamp(A), then Ti reads overwritten value of A. So, the read operation on data item A is rejected, and transaction Ti is rolled back.
  - If TS(Ti) > = W-timestamp(A), then the read operation on data item A is executed, and R-timestamp(A) is set to the maximum of R-timestamp (A) and TS (Ti) i.e. R-timestamp (A) = max(R-Timestamp(A),TS (Ti))
- Suppose that transaction Ti issues Write(A)
  - If TS (Ti) < R-timestamp (A), then the system rejects the write operation and Ti is rolled back. If TS (Ti) < W-timestamp (A), which means Ti is attempting to write an obsolete value of A. So, the write operation is rejected and Ti is rolled back.

- Otherwise, the Write (A) operation is executed.

Thomas' Write Rule

This rule states

- Suppose that transaction Ti issues Write(A).
  - If TS(Ti) < W-timestamp(A) and TS (Ti) < R-timestamp(A), then instead of rejecting the Write(A) operation it can be ignored and obsolete writes are deleted.

Q. 6(b) A transaction may be waiting for more time for an Exclusive (X) lock on an item, while a sequence of other transactions request and are granted as Shared (S) lock on the same item. What is this problem? How it is solved by two phase lock protocol?

(Ans. : Refer sections 4.8.2.1 and 4.8.2.2)

(Chap 4, 8 Marks)

Ans. :

Starvation : A transaction may be waiting for an X-lock on an item, while a sequence of other transactions request and are granted an S-lock on the same item. The same transaction is repeatedly rolled back due to deadlocks. Concurrency control manager can be designed to prevent starvation.

Concurrency control manager which grants the locks to transactions. The transactions requests for a lock to this concurrency control manager. The locking mechanism allows reading the data items simultaneously for various transactions but allows only one transaction to perform write operation on the data item at a time.

Lock Based Protocols

A. Two-Phase Locking (2PL)

- In this scheme, each transaction makes lock and unlock request in 2 phases :
  1. **A Growing Phase (or An Expanding Phase or First Phase) :** In this phase, new locks on the desired data item can be acquired but none can be released.
  2. **A Shrinking Phase (or Second Phase) :** In this phase, existing locks can be released but no new locks can be acquired.
- In the beginning, the transaction is in growing phase in which it acquire lock as per requirement. After completing the work, the transaction releases the locks and enters into shrinking phase.
- The transaction cannot request for new locks after releasing the locks.

T
Lock-X(A);
Read(A);
A = A + 500;
Write(A);
Lock-X(B);
Read(B);
B = B - 100;
Write(B);
Unlock(A);
Unlock(B);

- The point in the schedule where the transaction acquires its final lock is called as **lock point**. This point is the end of growing phase. The transactions can be ordered as per their lock points. This sequence of transactions is the serailizability ordering for the transactions. This protocol assures serializability.
- In serializability the main issue is regarding write operation. The parallel write operations may create inconsistency in the database. The parallel read operations do not create any problem.
- In serial schedule, as the transactions are executed one after other, there is no risk of parallel write operations which may lead to inconsistency.
- The database transactions can be controlled in 2-Phase locking mechanism by applying the exclusive lock. The exclusive (write) lock is released in shrinking phase.
- Unless the exclusive lock of a data item is released by the transaction, other transaction can acquire exclusive lock on that data item.
- This helps to maintain the serailizability. It is not sure that Two-phase locking is free from deadlocks.

- In two-phase locking, there is possibility of cascading roll-back. This can be avoided by using **strict two-phase locking** in which a transaction must hold all its exclusive locks till it commits/aborts.
- **Rigorous two-phase locking** is stricter. In it all the locks are held till commit/abort. Here transactions can be serialized in the order in which they commit.

B. Strict Two-Phase Locking

In Strict two phase locking protocol the first step of acquiring all the locks is same as first phase of two phase locking protocol. But the difference in Strict two phase locking and two phase locking is that Strict two phase locking protocol does not release the lock after using it. It holds all the locks till it does not reach to commit point. When it reached to the commit point it releases all locks at a time. Cascading rollback can be prevented by using the strict phase locking protocol. This protocol is used to make Cascadless schedules.

Q. 7(a) Explain speed-up and scale-up parameters of parallel systems. What are the different factors limiting the speed-up and scale-up parameters.

(Ans. : Refer sections 5.4.1 and 5.4.1.1)
(Chap 5, 8 Marks)

Ans. :

Speedup and Scaleup

To measure the performance of parallel processing we can use two important properties:

1. Speedup

- The extent in which more hardware can perform the same task in less time than the original system is known as **Speedup.** It means the execution of task done in less time by increasing degree of parallelism.
- The time which is required to process a task is inversely proportional to the time when number of resources are used.
- With good speedup, additional processors reduce system response time. We can measure speedup is as follows :

$$\text{Speedup} = \frac{\text{time\_original}}{\text{time\_parallel}}$$

Where,

- Time_Parallel is the time spent by a larger, parallel system on the given task
- For example, If the original system took 60 seconds to perform a task, and two parallel systems took 30 seconds, then the value of speedup would be 60/30 = 2

Linear Speedup

We can say that speedup is linear when speedup is equal to N. That is the elapsed time of small system is N times larger than the elapsed time of large system where N is the number of resources.

Sub-linear Speedup

When speedup is less than N then it is called as sub-linear speedup.

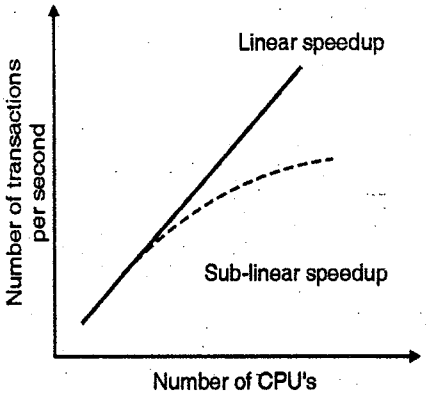


Fig. 1-Q. 7(a) : Speedup

2. Scaleup

- The ability to keep the same performance level when both workload and resources increase proportionally is known as **Scaleup.** It is the process of handling large task in same amount of time by increasing degree of parallelism.

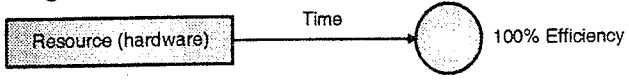
We can measure the ScaleUp is as follows:

$$\text{Speedup} = \frac{\text{Volume\_parallel}}{\text{Volume\_original}}$$

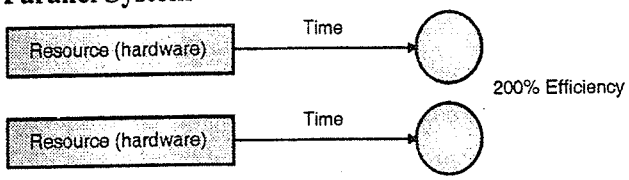
Where, Volume_Parallel is the transaction volume processed in a given amount of time on parallel system.

- In case of good Scaleup if there is increase in transaction volume, then to keep response time constant, we can add hardware resources like CPU.

Original System



Parallel System



Different Factors Affecting the Speedup and Scaleup Attributes

Startup costs

For all the parallel operations, there is a associated cost involved in starting the process. When a big process is break in small processes then it consumes some amount of time and recourses. If we want to execute a query in parallel then we need to partition the table and instructs various processors to execute the query in parallel.

Consider a query to sort the data.

```
Select * from emp order by ename
```

Now to execute this query in parallel we need to partition the table as per sorting criteria and instructs various processors to execute the query in parallel. Both of these operations need some amount of time before the parallel execution.

Interference

The various resources are shared by all he processes which executes in parallel. Whenever interference of a new process happens, it leads to slow down the overall access of all the processes. This affects both speed-up and scale-up.

Skew

It is difficult to break a big tasks in equal size small tasks. In such case the performance of the system depends upon the slowest CPU which processes the largest sub task. This type of uneven division of big tasks into smaller ones is called as skew. This also affects the speed-up and scale-up.

Q. 7(b) In both, Shared nothing parallel architecture and distributed system architecture resources are not shared, then how shared nothing parallel systems are different than distributed systems? Also explain in brief other parallel system architecture.

(Ans. : Refer section 5.4.2) **(Chap 5, 9 Marks)**

Ans. :

Shared Nothing Architecture

- The shared nothing architecture is a distributed computing architecture in which each node is independent. More specifically, none of the nodes share memory or disk storage.
- In this architecture each processor has its own local memory and local disk.
- Intercommunication channel is used by the processors to communicate.
- The processors can independently act as a server to serve the data of local disk.

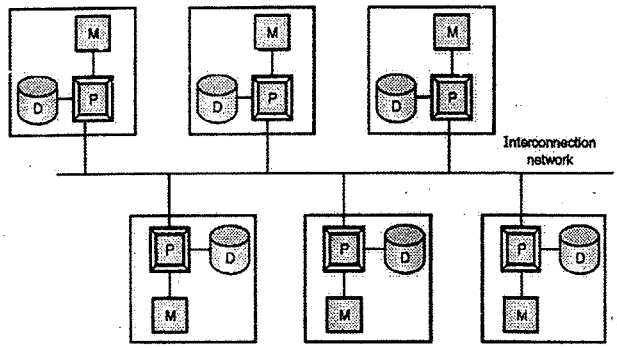


Fig. 1-Q. 7(b) : Shared nothing architecture

Distributed Database system

Distributed database is the type of database management system in which number of databases are stored at various locations and interconnected through the computer networks. Means we can say that, "Distributed database is a collection of various interconnected databases which are physically spread at various locations and communicate via a computer network. In distributed database system each site may have its own memory and its own Database Server.

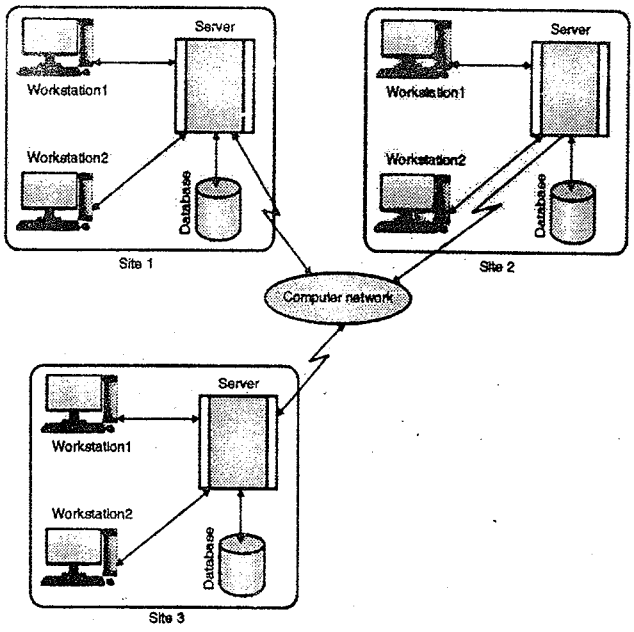


Fig. 2-Q. 7(b) : Distributed Database System

Parameter	Shared nothing architecture	Distributed system
Local memory	– In this architecture each processor has its own local memory and local disk.	In distributed database system each site may have its own memory and its own Database Server.
Communication	– Intercommunication channel is used by the processors to communicate.	Computer network is used for communication.
Advantages	1. This architecture is scalable regarding the number of processors. Increase in their number is easy and flexible. 2. When nodes get added transmission capacity increases.	1. Increases reliability 2. Improve performance 3. Increase availability
Disadvantages	1. The cost of communication is higher than shared memory and shared disk architecture. 2. The data sending involves the software interaction	1. Database design more complex. 2. Increased processing overhead

Other parallel system architecture

There are four different architectural models of parallel database :

1. Shared memory
2. Shared disk
3. Shared nothing
4. Hierarchical

1. Shared memory

In this architecture of parallel database common memory is shared among the multiple processors. These processors are connected through the interconnection network to the main memory and disk. The connection used in this architecture is usually high speed network connection which makes data sharing easy. Shared memory architecture have large amount of cache memory at each processors.

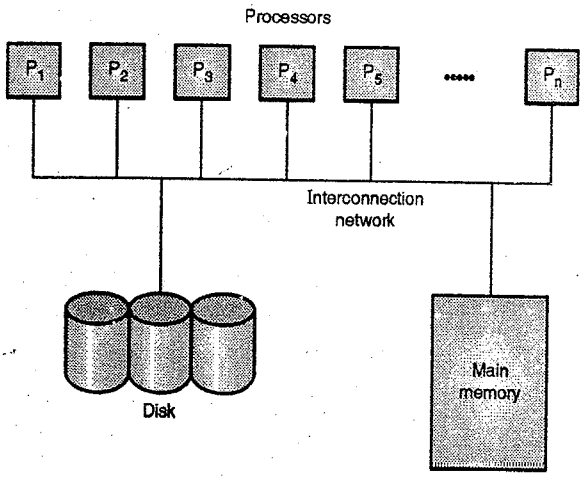


Fig. 3-Q. 7(b) : Shared memory architecture

Advantages of shared memory architecture

1. Any processor can access data easily.
2. Effective communication between processors through common memory address space.
3. Communication overheads are less.

Disadvantages of shared memory architecture

1. Waiting time of processors is increased due to more number of processors
2. Degree of parallelism is limited.
3. Addition of processor slows down the existing processors.
4. When a processor tries to access the data updated by other processors, then we have to take care that the data is of latest updated version.

2. Shared Disk Architecture

In the shared disk architecture of parallel database system single disk is shared among all the processors.

These all processors also have their own private memory which makes data sharing efficient.

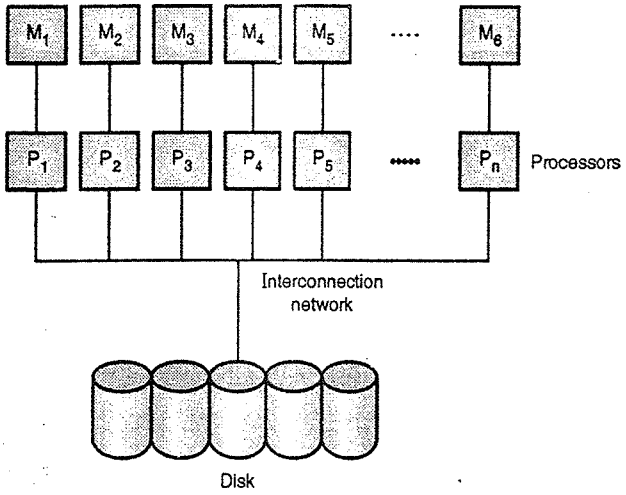


Fig. 4-Q. 7(b) : Shared disk architecture

Advantages of shared disk architecture

1. It has fault tolerance means failure of any processor does not lead to execution stop; rather any other processor completes the task.
2. As compared to shared memory architecture it supports large number of processors.
3. This architecture permits high availability.
4. Interconnection to the memory is more smooth and free.

Disadvantages of shared disk architecture

1. The scalability of Shared disc system is limited as large amount of data travels through the interconnection channel.
2. The speed of existing processors may slow down if more processors get added.

3. Hierarchical Architecture

Hierarchical model architecture is also known as Non Uniform Memory Architecture. This is nothing but the combination of shared disk, shared memory and shared nothing architecture.

Consider there are two groups of processors say A and B. All processors from both the groups have local memory. But processors from other groups can access memory which is associated with the other group in coherent.

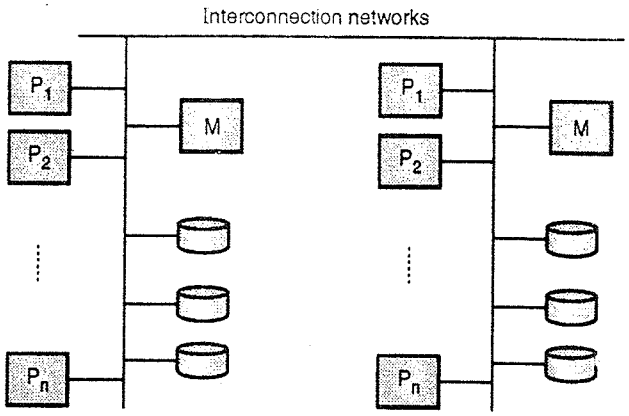


Fig. 5-Q. 7(b): Hierarchical architecture

Advantages of Hierarchical Architecture

1. The availability of memory is more in this architecture.
2. The scalability of system is also more.

Disadvantages of Hierarchical Architecture

1. This architecture is costly as compared to other architectures

OR

Q. 8(a) How two phase commit protocol to ensure the atomicity in distributed transaction, handles the following failures:

i) Failure of participating site

ii) Failure of coordinator

iii) Failure due to network partition

(Chap 5, 8 Marks)

Ans. :

(i) Failure of participating site

- If the coordinator Ci finds that a site has failed, it takes the following actions.
- If the site fails before responding with a ready T message to Ci, it is assumed to have responded with an abort T message.
- If the site fails after the coordinator has received the ready T message from the site, the rest of the commit protocol is executed in the normal fashion, ignoring the failure of the site.
- Consider each of the possible cases:

  i) The log contains a <commit T> record. In this case, the site executes redo (T).

ii) The log contains an <abort T> record. In this case, the site executes undo (T).

iii) The log contains a <ready T> record. In this case, the site must consult Ci to determine the fate of T.

- If C_i is up, it notifies S_k regarding whether T committed or aborted. In the former case, it executes redo (T); in the latter case it executes undo (T).
- If C_i is down, S_k must try to find the fate of T from other sites. It does so by sending a query status T message to all the sites in the system. On receiving such a message, a site must refer its log to determine whether T has executed there and if T has, whether T committed or aborted.
- It then notifies S_k about this outcomes. If no site has the appropriate information, then S_k can neither abort nor commit T.
- The decision concerning T is postponed until S_k can obtain the needed information. Thus, S_k must periodically resend the query-status message to the other sites. It continues to do so until a site recovers that contains the needed information.

(ii) Failure of the coordinator

- If the coordinator fails in between the execution of the commit protocol for transaction T, the participating site must decide the fate of T.
- In certain cases, the participating sites cannot decide whether to abort or commit T and therefore these sites must wait for the recovery of the failed coordinator.

i) If an active site contains an <abort T> record in its log, then T must be aborted.

ii) If an active site contains a <commit T> record in its log, then T must be committed.

iii) If some active site does not contain a <ready T> record in its log, then the failed coordinator C_i cannot have decided to commit T because a site that does not have a <ready T> record in its log cannot have sent a ready T message to C_i.

- However, the coordinator may have decided to abort T. Rather than wait for C_i to recover, it is preferable to abort T.

(iii) Failure due to Network partition

When network is partitioned, two possibilities are exists

i) The coordinator and all its participants remain in one partition. In this case, the failure has no effect on the commit protocol.

ii) The coordinator and its participants belong to several partitions. From the viewpoint of the sites in one of the partitions, it appears that the sites in other partitions have failed. Sites that are not in the partition containing the coordinator simply execute the protocol to deal with failure of the coordinator. The coordinator and the sites that are in the same partition as the coordinator follows the usual commit protocol, assuming that the sites in the other partitions have failed.

Q. 8(b) For concurrency control in distributed transaction distributed lock manager approach is used, explain in detail different approaches for dealing with replication of data items in distributed lock manager approach.

(Ans. : Refer section 5.8) **(Chap 5, 9 Marks)**

Ans. :

Concurrency Control in Distributed Database

In distributed database systems, database is typically used by many users. These systems usually allow multiple transactions to run concurrently i.e. at the same time.

The activity of coordinating concurrent access to a single database in a multiuser database management system (DBMS) is known as **Concurrency control**. It allows users to access a database in a multi-programmed fashion. In centralized database system, several locking protocols are used for concurrency control. The major difference between centralized and distributed database system is that how lock manager deal with replicated data.

There are some approaches for concurrency control in distributed database :

A) Single Lock Manager

- In distributed database system which have several sites, maintains a single lock manager at a particular chosen site this concept is known as single lock manager approach. Fig. 1-Q. 8(b) shows the single lock manager approach.
- Here the sites S1, S2, S3, S4, S5, and S6 are present among those the data D is replicated on S1 S6 and S4. And the site S3 acts as a Lock Manager.
- At the site S2 transaction T1 requests for data item D (step 1). These requests are forwarded by the site S2 to the lock manager's site S3 for granting purpose (step 2).

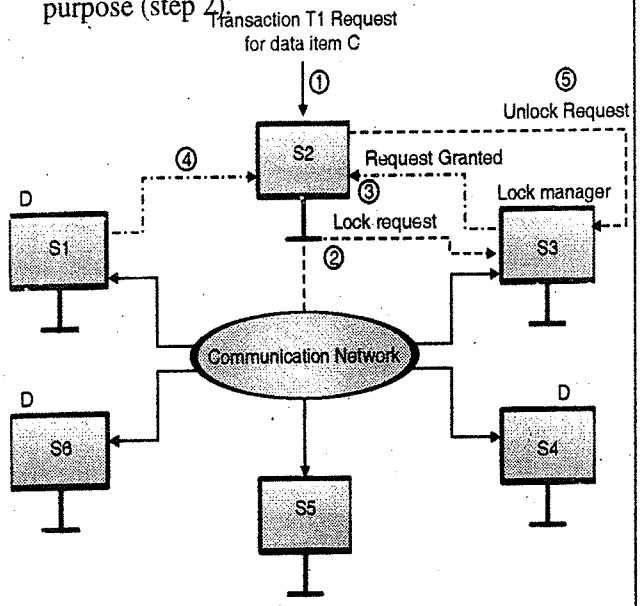


Fig. 1-Q. 8(b) : Single lock manager approach

- If the requested data item is free then the request for that data item is granted immediately by the lock manager (step 3).
- If the request is granted then the data item can be read from any site where the replica of required data item is present; here, the site S2 takes data item D from site S1(step 4).
- After the transaction T1 executed successfully the Transaction /manager at site S2 again send the request for unlock the data item so that other transactions can used the data item D now (step 5).

B) Distributed Lock Manager

- In distributed database system there are several sites and data is replicated or fragmented on those sites. The distributed lock manager approach is nothing but the distributing the functionality of lock manager over several sites.
- Fig. 2-Q. 8(b) shows the distributed lock manager approach using primary copy protocol.
- Here the sites S1, S2, S3, S4, S5, and S6 are present. Data A, B, C is replicated on multiple sites.
- Among the six sites S6 holds primary copy of data item B so the lock manager for granting data access on B is present at only site S6, similarly S5 site acts as a lock manger for data item A and S4 acts as a lock manager for data item C.

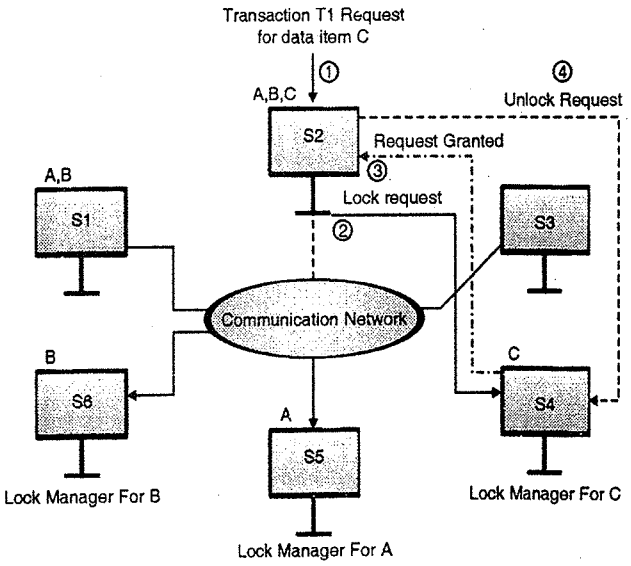


Fig. 2-Q. 8(b) : Distributed lock manager approach

- At the site S2 transaction T1 requests for data item C (step 1). Even if the replica of C is present at site S2 locally it is mandatory to first the request lock for accessing C from site S4 as it is acts as lock manager for C. So the transaction manager at site S2 sends lock request to site S4(step 2).
- If the requested data item is free then the request for that data item is granted immediately by the lock manager, hence site S4 grants lock request for C(step 3).

- Now the transaction T1 can access local copy of Data item C present at site S2(step 4).
- After the transaction T1 executed successfully the Transaction manager at site S2 again send the request for unlock the data item so that other transactions can used the data item C now(step 5).

Q. 9(a) BASE Transactions ensures the properties like Basically Available, Soft State, Eventual Consistency. What is soft state of any system, how it is depend on Eventual consistency property ?

(Ans. : Refer section 6.6.2) **(Chap 6, 8 Marks)**

Ans. :

- What ACID means in traditional RDBMS community before moving to the BASE Properties.
- The four properties are as follows.
- **Atomicity :** This property states that, either all operations contained by a transaction are done successfully or none of them complete at all.
- **Consistency :** The consistency property ensures that the transaction executed on the database system will bring the database from one valid state to another.
- **Isolation :** In case of concurrent transactions, the isolation property ensures that the system state should be same that would be obtained if transactions were executed sequentially.
- **Durability :** The durability property assures that after transaction committed successfully the updates made should remain permanent in the database even in the event of power loss, crashes, or errors.

ACID provides *strong consistency* (synchronous transactions) for partitioned databases and thus provides less availability. Consistency is always preferable, but it is not always available.

BASE Properties

- **Base** Property means **B**asically **A**vailable, **S**oft state, **E**ventual consistency.
- Consistency is always preferable, but it is not always available. In the NoSQL world, ACID transactions are less suitable as some DBMS do not have requirements for strict consistency, data freshness and accuracy in order to gain other benefits, like scale and resilience. By considering this, the BASE Consistency model is developed.
- The BASE properties are as follows :

1. **Basic Availability :** Most of the time the database appears to work.
2. **Soft-state :** It is not necessary that the stores should be write-consistent. Also the different replicas have to be mutually consistent all the time, hence it is depends on Eventual consistency property.
3. **Eventual consistency :** Stores may show the consistency at some later point. Eventual consistency which is normally asynchronous in nature is a form of a weaker consistency which improves speed and availability.

- The BASE (Basically Available, Soft state, Eventual consistency) is the opposite of ACID.
- ACID is pessimistic i.e. consistency is required at the end of every operation. BASE is optimistic, i.e. it accepts that there is uncertainty in consistency.
- A BASE Model focus on availability as it is important for scale, but it doesn't offer guaranteed consistency of replicated data at write time.
- At the end we can say the BASE model of consistency provides a less strict assurance than **ACID :** Data will be consistent in the future, either at read time or may be always consistent for some points .

Q. 9(b) List the different NOSQL data Models. Explain document store NOSQL data model with example.

(Ans. : Refer sections 6.2 and 6.2.2) **(Chap 6, 8 Marks)**

Ans. :

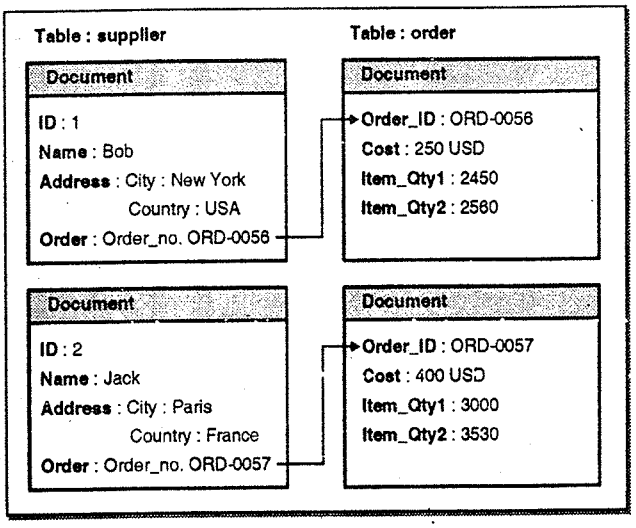
Types of NoSQL Database

There have been various approaches to classify NoSQL databases, each with different categories and subcategories.

Following are some types of NoSQL :

1. Key Value Store
2. Document Store
3. Column Store
4. Graph

Document Store



- These databases store records as "documents" where a document can generally be thought of as a grouping of key-value pairs.
- The documents are identified by the unique keys which represents them. One defining characteristics of a document-oriented database is that in addition to the key lookup performed by a key-value store, the database offers an API or query language that retrieves documents based on their contents.
- Document databases are extension to key-value store. Addition to query capabilities of key-value databases, they provide indexing and the ability to filter documents based on attributes in the document.
- Examples of Document Store NoSQL database applications :
  - **MongoDB :** This system provides operational capabilities for interactive, real-time workloads where data is primarily captured and stored.
  - Couchbase

Advantages of Document Store

- The performance is good and the distribution across various servers becomes a lot easier.
- There is no need of translation between object in SQL and application. The object can directly be converted into document.
- "They have strong indexing features and can rapidly execute different queries.

OR

Q. 10(a) Explain how NOSQL databases are different than relational databases? **(Chap 6, 8 Marks)**

Ans. :

Parameters	NoSQL	Relational database
Data structure flexibility	The data structures used by NoSQL databases are more flexible than relational database system.	The data structures used by Relational database is less flexible than NoSQL database system.
Also known as	Nosql database primarily known as non Relational Databases.	SQL databases are primarily known as Relational Databases.
Designed for	NoSQL databases are designed to handle unstructured data	Relational databases are designed to handle structured data.
scaling	To scale a NoSQL database is cheaper than a relational database.	To scale a relational database is expensive than a NoSQL database.
Development model	NoSQL databases are open source.	Relational databases are closed source.

Q. 10(b) Write short note on Hadoop : HDFS, MapReduce. *(Ans. : Refer section 6.11)* **(Chap 6, 8 Marks)**

Ans. : Hadoop

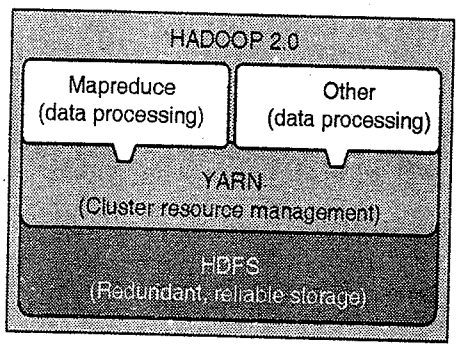
- **Hadoop** is an open source, Java-based programming framework which supports the processing and storage of extremely large sets of data in a distributed computing environment using simple programming models.
- Hadoop has very strong processing power and the ability to handle virtually unlimited parallel tasks.
- With the help of Hadoop, applications can be run on systems with thousands of commodity hardware nodes. It can handle thousands of terabytes of data. Hadoop has distributed file system which facilitates rapid data transfer rates among nodes.
- This allows the system to proceed even in case one or more nodes get failed. This approach avoids the unexpected data loss.
- Hadoop has quickly emerged as a foundation for big data processing tasks like scientific analytics of data, planning of business and sales, and processing enormous volumes of data including social media data.

History of Hadoop

- Computer scientists Doug Cutting and Mike Cafarella created Hadoop in 2006 to support distribution for the Nutch (search engine).
- The main aim is to increase the speed of search results by the distribution of data and implement calculations on different computers by multitasking.
- Later on Cutting joined Yahoo but him still works on the Nutch project with the ideas based on Google's early work with automating distributed data storage and processing.
- The Nutch Prject divided into two parts –
  - Nutch - Web crawler portion
  - Hadoop - Distributed computing and processing portion
- In 2008, Yahoo released Hadoop as an open-source project. Now Apache Software Foundation (ASF) manages the Hadoop's framework and ecosystem of technologies.

Modules (Components) of Hadoop

HDFS : HDFS stands for Hadoop Distributed File System. It states that the files will be broken into blocks and stored in nodes over the distributed architecture. It provides high-throughput access to application data.



- **Yarn :** Yarn stands for "Yet another Resource Negotiator". It is used for job scheduling and managing the cluster (multiple nodes).
- **Map Reduce :** This is YARN-based system for parallel processing of large data set using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair.
- **Hadoop Common :** These Java libraries and utilities are used to start Hadoop. These are used by other Hadoop modules. These libraries provide file system and OS level abstractions.

Advantages of Hadoop

1. Huge amounts of any kind of data can be stored and processed quickly.
2. **Computing power :** Hadoop's distributed computing model processes big data fast.
3. **Fault Tolerance :** In case of failure of any node the tasks are automatically redirected to other nodes.
4. **Flexibility :** Any kind of unstructured data like text, images and videos can be stored.
5. **Low cost :** This open-source framework is free.
6. **Scalability :** New nodes can be easily added to handle big tasks.

Disadvantage of Hadoop

1. Hadoop is rough in manner because the software is under active development.
2. Programming model is very restrictive.
3. Joins of multiple datasets are tricky and slow.
4. **Cluster management is hard :** In the cluster, operations like debugging, distributing. Software, collection logs etc are too hard.
5. Requires care and may limit scaling.

MapReduce

- MapReduce is an important part of Hadoop. It is a software framework which is used to write applications easily to process huge amount of data (multi-terabyte data-sets) simultaneously on large clusters (thousands of nodes) in reliable, fault-tolerant manner.
- MapReduce basically referes to two tasks performed by the Hadoop programs. One is map and another is reduce.

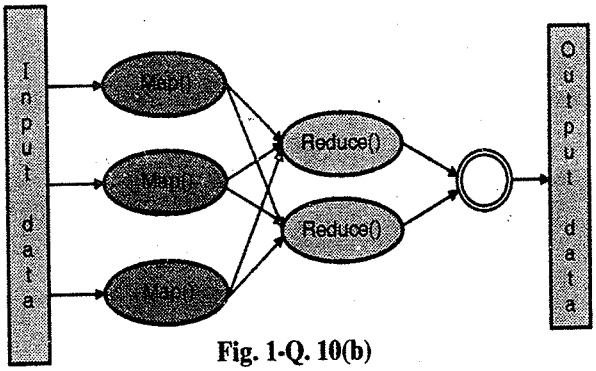


Fig. 1-Q. 10(b)

- Hadoop programs perform following two tasks on **MapReduce :**
  - **The Map Task :** This is the first task, which takes a set of data and converts it into another set of data in which individual elements are broken down into tuples (key/value pairs).
  - **The reduce :** This job takes the output of previously executed map task as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.
- In MapReduce framework, the data of input and output is stored in a file system. The framework handles the scheduling of all the tasks, monitoring these tasks and if fails, re-executes them.
- The main advantage of MapReduce is that it is simple to scale data processing over multiple computing nodes. The data processing primitives are known as mappers and reducers in the MapReduce model.
- The MapReduce framework consists of single master JobTracker and one slave TaskTracker per cluster-node.
- **Master JobTracker :** The tasks under master are :
  - Managing the resources.
  - Tracking consumption and availability of resources.
  - Scheduling the jobs component tasks on the slaves.
  - Monitoring the tasks and re-executing the failed tasks.
- **The slaves TaskTracker :** It execute the tasks as per the directions of the master and provide task-status information to the master periodically.
- The JobTracker is very important in Hadoop MapReduce service. If JobTracker goes down, all running tasks get halted.

Advantages of MapReduce

1. Scalable.
2. Fault tolerant.
3. Simple coding model.
4. Supports unstructured data.

Hadoop Distributed File System (HDFS)

- The HDFS is the primary storage system used by Hadoop applications. HDFS is a distributed file system and a framework provided by Hadoop for the analysis and transformation of huge data sets which uses the MapReduce paradigm.

- The HDFS is based on Google File System (GFS). It provides high-performance access to data across Hadoop clusters (thousands of computers), HDFS has become a key tool for managing pools of big data and supporting big data analytics applications.
- HDFS is usually deployed on commodity hardware of low-cost where the possibility of server failures is common. The file system is designed to be highly fault-tolerant.
- The HDFS facilitates the rapid transfer of data between different computer nodes and enables Hadoop systems to proceed its execution even if one or more nodes get failed. That decreases the risk of catastrophic failure, even in the event that numerous nodes fail.

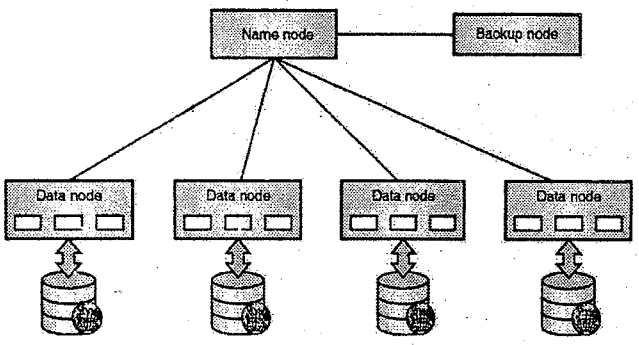


Fig. 2-Q. 10(b)

- The architecture used by HDFS is known as master/slave architecture.
- NameNode which manages the metadata of file system and DataNode which stores the actual data.
- The HDFS namespace is a hierarchy of files and directories. Inodes are used to represent these file and directories.
- Inodes are used to record attributes such as permissions, modification and access times etc. The file content is split into large blocks and each block of the file is independently replicated at multiple DataNodes.
- The tree structure of namespace is maintained by the NameNode. It maps the blocks to DataNodes.
- In a cluster there may be hundreds of DataNodes and thousands of HDFS clients per cluster, as number of application tasks can be executed by each DataNode simultaneously.

Advantages of HDFS

1. High scalability.
2. Low limitation.
3. Open source.
4. Low cost.

Disadvantages of HDFS

1. Still rough - means software under active development.
2. Programming model is very restrictive.
3. Cluster management is high.

❑❑❑