

Marathwada Mitra Mandal's
COLLEGE OF ENGINEERING, PUNE
Karvenagar,Pune-411 052



Department of Computer Engineering

Lab Manual

Database Management System Lab

310246

Prepared by,

Dr. Asma Shaikh

Dr. Kalpana Thakre

Ms.Kirti Satpute

TE COMP

Semester I Academic Year 2023-24

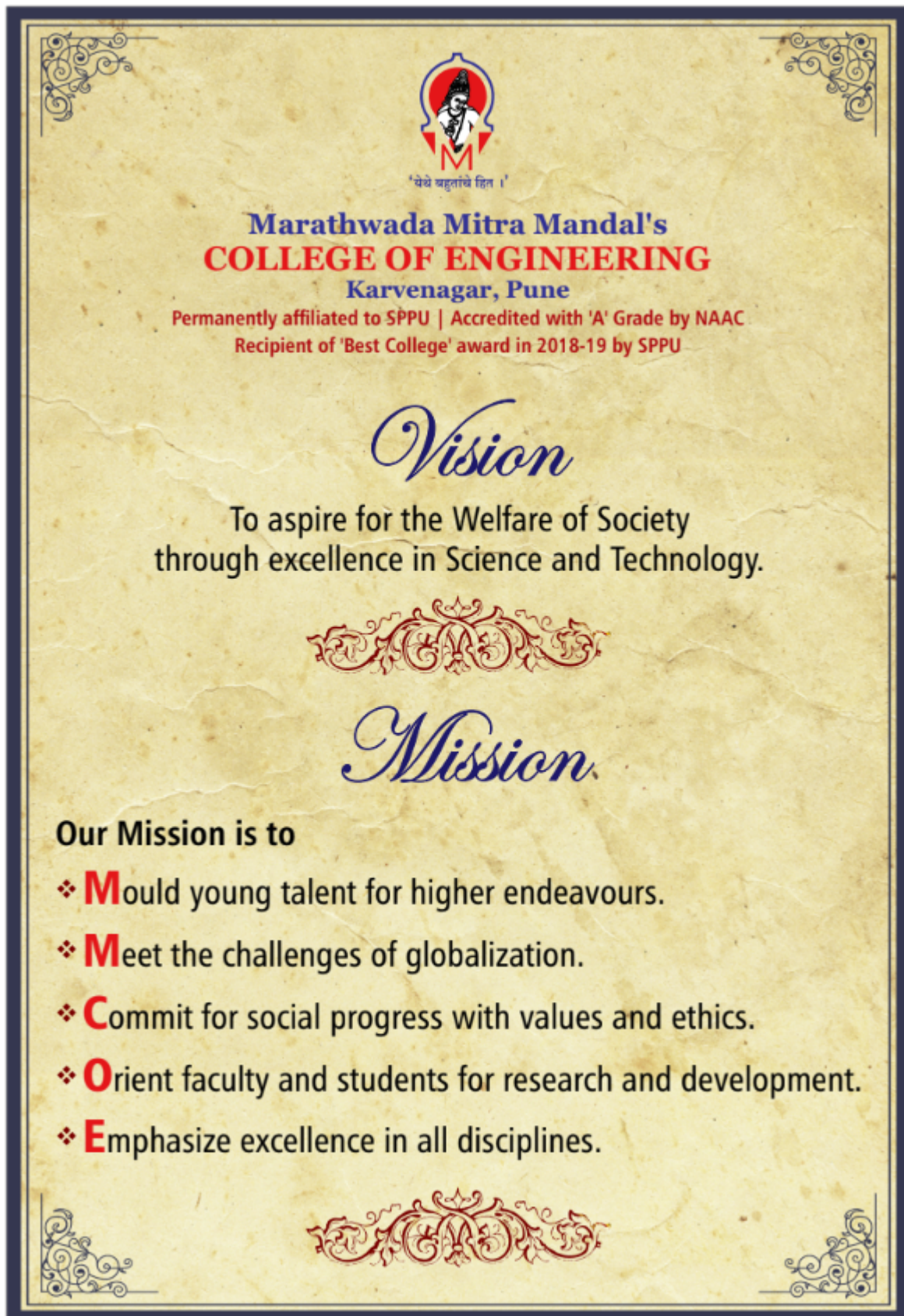
Preface

Database Management System (DBMS) is software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs that manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software store and retrieve data.

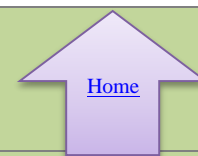
DBMS allows users to create their own databases as per their requirements. The term "DBMS" includes the user of the database and other application programs. It provides an interface between the data and the software application.

Here are the characteristics and properties of a Database Management System:

- Provides security and removes redundancy
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing
- Database Management Software allows entities and relations among them to form tables.
- It follows the ACID concept (Atomicity, Consistency, Isolation, and Durability).
- DBMS supports a multi-user environment that allows users to access and manipulate data in parallel.



Savitribai Phule Pune University
Third Year of Computer Engineering (2019 Course)
310246:Database Management Systems Laboratory



Teaching Scheme
Practical: 04 Hours/Week

Credit Scheme: 02

Examination Scheme and Marks
Term work: 25 Marks
Practical: 25 Marks

Companion Course: Database Management Systems (310241)

Course Objectives:

- To develop Database programming skills
- To develop basic Database administration skills
- To develop skills to handle NoSQL database
- To learn, understand and execute process of software application development

Course Outcomes:

On completion of the course, learners will be able to

- CO1:** Design E-R Model for given requirements and convert the same into database tables
- CO2:** Design schema in appropriate normal form considering actual requirements
- CO3:** Implement SQL queries for given requirements , using different SQL concepts
- CO4:** Implement PL/SQL Code block for given requirements
- CO5:** Implement NoSQL queries using MongoDB
- CO6:** Design and develop application considering actual requirements and using database concepts

Guidelines for Instructor's Manual

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

Guidelines for Student's Laboratory Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of Certificate, table of contents, and handwritten write-up of each assignment (Title, Date of Completion, Objectives, Problem Statement, Software and Hardware requirements, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, Test Data Set(if applicable), mathematical model (if applicable), 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 must be avoided. Use of DVD containing students programs maintained by Laboratory In-charge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory.

Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work should be based on overall performance of Laboratory assignments by a student. Each Laboratory assignment assessment will assign grade/marks based on parameters, such as timely completion, performance, innovation, efficient codes, and punctuality.

Guidelines for Practical Examination

Problem statements must be decided jointly by the internal examiner and external examiner. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement. Relevant questions may be asked at the time of evaluation to test the student's understanding of the fundamentals, effective and efficient implementation. This will encourage, transparent evaluation and fair approach, and hence 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 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. Use of open source software is encouraged. 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.

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

Programming tools recommended: - MYSQL/Oracle, MongoDB, ERD plus, ER Win

Virtual Laboratory:

- <http://vlabs.iitb.ac.in/vlabs-dev/labs/dblab/labs/index.php>

Suggested List of Laboratory Experiments/Assignments

Assignments from all Groups (A, B, C) are compulsory

Sr. No.	Group A: SQL and PL/SQL
1.	<p>ER Modeling and Normalization:</p> <p>Decide a case study related to real time application in group of 2-3 students and formulate a problem statement for application to be developed. Propose a Conceptual Design using ER features using tools like ERD plus, ER Win etc. (Identifying entities, relationships between entities, attributes, keys, cardinalities, generalization, specialization etc.) Convert the ER diagram into relational tables and normalize Relational data model.</p> <p>Note: Student groups are required to continue same problem statement throughout all the assignments in order to design and develop an application as a part Mini Project. Further assignments will be useful for students to develop a backend for system. To design front end interface students should use the different concepts learnt in the other subjects also.</p>
2.	<p>SQL Queries:</p> <p>a. Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.</p> <p>b. Write at least 10 SQL queries on the suitable database application using SQL DML statements.</p> <p>Note: Instructor will design the queries which demonstrate the use of concepts like Insert, Select, Update, Delete with operators, functions, and set operator etc.</p>
3.	<p>SQL Queries - all types of Join, Sub-Query and View:</p> <p>Write at least 10 SQL queries for suitable database application using SQL DML statements.</p> <p>Note: Instructor will design the queries which demonstrate the use of concepts like all types of Join, Sub-Query and View</p>

4.	<p>Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory.</p> <p>Suggested Problem statement: Consider Tables:</p> <ol style="list-style-type: none"> 1. Borrower(Roll_no, Name, DateofIssue, NameofBook, Status) 2. Fine(Roll_no,Date,Amt) <ul style="list-style-type: none"> • Accept Roll_no and NameofBook 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 and 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. • Also handles the exception by named exception handler or user define exception handler. <p style="text-align: center;">OR</p> <p>Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.</p> <p>Note: Instructor will frame the problem statement for writing PL/SQL block in line with above statement.</p>
5.	<p>Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.</p> <p>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.</p> <p>Write a PL/SQL block to use procedure created with above requirement.</p> <p style="text-align: center;">Stud_Marks(name, total_marks) Result(Roll,Name, Class)</p> <p>Note: Instructor will frame the problem statement for writing stored procedure and Function in line with above statement.</p>
6.	<p>Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)</p> <p>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.</p> <p>Note: Instructor will frame the problem statement for writing PL/SQL block using all types of Cursors in line with above statement.</p>

7.	<p>Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).</p> <p>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.</p> <p>Note: Instructor will Frame the problem statement for writing PL/SQL block for all types of Triggers in line with above statement.</p>
8.	<p>Database Connectivity:</p> <p>Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)</p>
Group B: NoSQL Databases	
1.	<p>MongoDB Queries:</p> <p>Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators etc.).</p>
2.	<p>MongoDB - Aggregation and Indexing:</p> <p>Design and Develop MongoDB Queries using aggregation and indexing with suitable example using MongoDB.</p>
3.	<p>MongoDB - Map reduces operations:</p> <p>Implement Map reduces operation with suitable example using MongoDB.</p>
4.	<p>Database Connectivity:</p> <p>Write a program to implement MongoDB database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)</p>
Group C: Mini Project	
1.	<p>Using the database concepts covered in Group A and Group B, develop an application with following details:</p> <ol style="list-style-type: none"> Follow the same problem statement decided in Assignment -1 of Group A. Follow the Software Development Life cycle and other concepts learnt in Software Engineering Course throughout the implementation. Develop application considering: <ul style="list-style-type: none"> Front End : Java/Perl/PHP/Python/Ruby/.net/any other language Backend : MongoDB/MySQL/Oracle Test and validate application using Manual/Automation testing. Student should develop application in group of 2-3 students and submit the Project Report which will consist of documentation related to different phases of Software Development Life Cycle: <ul style="list-style-type: none"> Title of the Project, Abstract, Introduction Software Requirement Specification Conceptual Design using ER features, Relational Model in appropriate Normalize form Graphical User Interface, Source Code Testing document Conclusion. <p>Note:</p> <ul style="list-style-type: none"> Instructor should maintain progress report of mini project throughout the semester from project group Practical examination will be on assignments given above in Group A and Group B only Mini Project in this course should facilitate the Project Based Learning among students

@The CO-PO Mapping Matrix

PO/CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	-	1	3	-	3	1	1	1	3	1	-	1
CO2	2	2	3	-	2	-	1	-	3	-	1	-
CO3	-	1	2	-	2	1	-	1	3	-	-	2
CO4	-	1	2	-	2	-	-	-	3	2	1	-
CO5	-	1	2	-	2	-	2	-	3	1	-	1
CO6	2	2	3	-	3	1	-	-	3	-	2	1



Marathwada Mitra Mandal's
COLLEGE OF ENGINEERING
Department of Computer Engineering



Vision:

To contribute to welfare of society by empowering students with latest skills, tools and technologies in the field of computer engineering through excellence in education and research

Mission:

1. To provide Excellent Academic Environment for continuous improvement in the domain knowledge of Computer Engineering to solve real world problems.
2. To impart education to students with innovative and research skills to make them competent to solve social problems.
3. To strengthen links with industries through partnerships and collaborative developmental works.

Programme Outcomes:

Students are expected to know and be able –

1. To apply knowledge of mathematics, science, engineering fundamentals, problem solving skills, algorithmic analysis and mathematical modeling to the solution of complex engineering problems.
2. To analyze the problem by finding its domain and applying domain specific skills
3. To understand the design issues of the product/software and develop effective solutions with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.
4. To find solutions of complex problems by conducting investigations applying suitable techniques.
5. To adapt the usage of modern tools and recent software.
6. To contribute towards society by understanding the impact of Engineering on global aspect.
7. To understand environment issues and design a sustainable system.
8. To understand and follow professional ethics.
9. To function effectively as an individual and as member or leader in diverse teams and interdisciplinary settings.
10. To demonstrate effective communication at various levels.
11. To apply the knowledge of Computer Engineering for development of projects, and its finance and management.
12. To keep in touch with current technologies and inculcate the practice of lifelong learning.

Course Objectives:

- 1) To develop Database programming skills
- 2) • To develop basic Database administration skills
- 3) • To develop skills to handle NoSQL database
- 4) • To learn, understand and execute process of software application development

Course Outcomes:

On completion of the course, learners will be able to

- 1) Design E-R Model for given requirements and convert the same into database tables
- 2) Design schema in appropriate normal form considering actual requirements
- 3) Implement SQL queries for given requirements, using different SQL concepts
- 4) Implement PL/SQL Code block for given requirements
- 5) Implement NoSQL queries using MongoDB
- 6) Design and develop application considering actual requirements and using database concepts

Course Outcome	Program outcomes											
	1	2	3	4	5	6	7	8	9	1	1	12
310246.1	-	1	3	-	3	1	1	1	3	1	-	1
310246.2	2	2	3	-	2	-	1	-	3	-	1	-
310246.3	-	1	2	-	2	1	-	1	3	-	-	2
310246.4	-	1	2	-	2	-	-	-	3	2	1	-
310246.5	-	1	2	-	2	-	2	-	3	1	-	1
310246.6	2	2	3	-	3	1	-	-	3	-	2	1

INDEX

Sr. No.	Group	Asg. No.	Title of Assignment	CO	PO
1	A	1	Study of ER diagram and conversion of ER diagram into tables in normalized form	CO1	PO2,PO3, PO5, PO6,PO7,PO8, PO9,PO10,PO12
2		2A	Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym	CO1 CO2 CO3	PO2,PO3,PO5,PO6 ,PO7,PO8,PO9, PO10,PO11,PO12
		2B	Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operators.	CO1 CO2 CO3	PO2,PO3,PO5,PO6 ,PO7,PO8,PO9, PO10,PO11,PO12
3		3	Design at least 10 SQL queries for suitable database application using SQL DML statements:all types of Join, Sub-Query and View.	CO1 CO2 CO3	PO2,PO3,PO5,PO6 ,PO7,PO8,PO9, PO10,PO11,PO12
4		4	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) <ul style="list-style-type: none"> 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 After submitting the book, status will change from I to R. If the condition of fine is true, then details will be stored into a fine table. 	CO3 CO4	PO2,PO3,PO5,PO6 ,PO8,PO9,PO10,P O11

5		5	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.	CO3 CO4	PO2,PO3,PO5,PO6 ,PO8,PO9,PO10,P O11
6		6	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)	CO3 CO4	PO2,PO3,PO5,PO6 ,PO8,PO9,PO10,P O11
7		7	Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on the 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 the Library_Audit table. Frame the problem statement for writing Database Triggers of all types, in-line with the above statement. The problem statement should clearly state the requirements.	CO3 CO4	PO2,PO3,PO5,PO6 ,PO8,PO9,PO10,P O11
8		8	Database Connectivity: write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)	CO1 CO2 CO3 CO6	PO1,PO2,PO3, PO5,PO6,PO7,PO8 ,PO9, PO11,PO12
9	B	9	Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)	CO5	PO2,PO3,PO5, PO7,PO9,PO10, PO12
10		10	Implement aggregation and indexing with suitable examples using MongoDB.	CO4	PO2,PO3,PO5, PO7,PO9,PO10, PO12
11		11	Implement Map reduce operation with suitable examples using MongoDB.	CO4	PO2,PO3,PO5, PO7,PO9,PO10, PO12
12		12	Database Connectivity: Write a program to implement MongoDB database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.).	CO5 CO6	PO2,PO3,PO5, PO6,PO7,PO9, PO10,PO11,PO12
13	C	13	Mini Project- Using the database concepts covered in Group A and Group B, develop an software application :	CO1 CO2 CO3 CO5 CO6	PO1,PO2,PO3, PO5,PO6,PO7, PO8,PO9,PO10. PO11,PO12
14	Content Beyond Syllabus	14	Design star schema of Mini Project using ERDPlus (Content Beyond Syllabus)	CO1 CO2	PO2,PO3, PO5, PO6,PO7,PO8, PO9,PO10,PO12

15	VLabs	15	Data Definition Language(DDL) Statements: (Create table, Alter table, Drop table)	CO1 CO2 CO3	PO2,PO3,PO5,PO6 ,PO7,PO8,PO9, PO10,PO11,PO12
----	-------	----	---	-------------------	--

CERTIFICATE

Certified that Mr./Ms._____ **of Class TE Computer Engineering Roll No.**_____ **has completed the Journal assignments in the subject DBMS Lab during the academic year 2023-24 Sem I.**

Signature of the Faculty

Signature of Head of the Department

Date:

Software Required:

- 1 64 bit open source operating system- Ubuntu
1. MySQL
2. Oracle 11g Express Edition(Ubuntu 16.04TLS)
3. MongoDB
4. Java

Write-ups must include:

- Group
- Assignment No.
- Aim
- Problem Statement
- Prerequisites
- Course Objectives
- Course Outcomes
- Theory(in brief)
- Test Cases
- Conclusion
- FAQs:
- Output: Soft copy of program with output.

Department of Computer Engineering, MMCOE

GROUP: A**ASSIGNMENT NO: 1**

Aim: Case Study of ER diagram for (Hospital Management System)

Note: Use your mini project statement for drawing ER diagram

Problem Statement:

Study of ER diagram and conversion of ER diagram into RDBMS

Course Objectives:

1. To develop basic, intermediate and advanced Database programming skills
2. To develop basic database administration skills

Course Outcome:

- 1) Ability to handle database- MySQL
- 2) Design ER model and convert ER diagram into database tables

Theory:**ER Diagram**

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

ER Diagrams Symbols & Notations

Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

Following are the main components and its symbols in ER Diagrams:

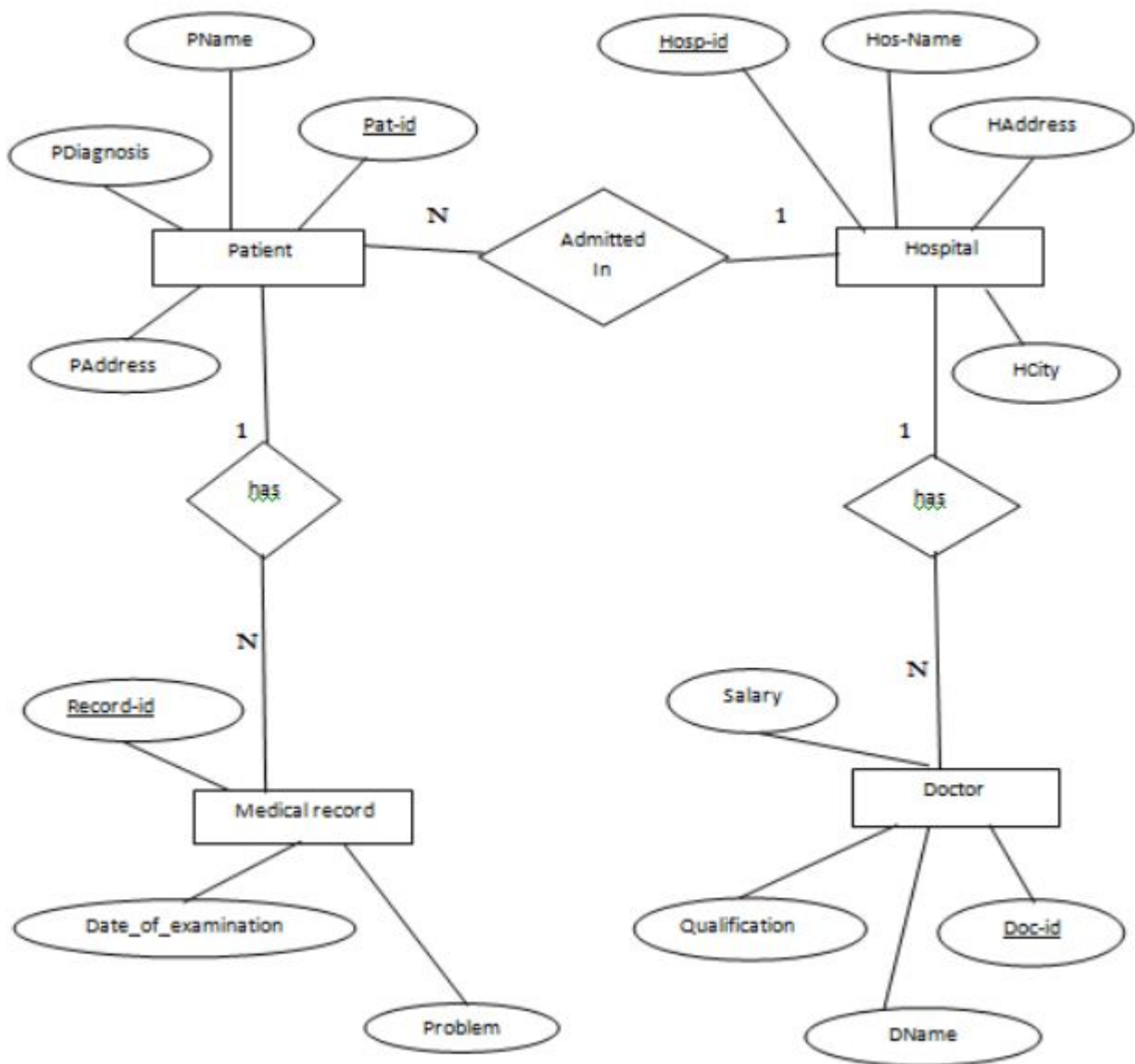
- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types

Department of Computer Engineering, MMCOE

- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes



Step 1: E-R Diagram



Step 2: Converting the E-R Diagram into Tables

b. Converting entity to table and attribute to columns

Hospital

Hosp-id	Primary Key
HCity	
HAddress	
Hos-Name	
Pat-id	Foreign key references to Pat-id of Patient table
Doc-id	Foreign key references to Doc-id of Doctor table

Department of Computer Engineering, MMCOE

Patient

Pat-id	Primary Key
PName	
PAddress	
PDagnosis	
Record-id	Foreign key references to Record-id of Medical Record table
Hosp-id	Foreign key references to Hosp-id of Hospital table

Medical Record

Record-id	Primary Key
Problem	
Date_of_exam ination	
Pat-id	Foreign key references to Pat-id of Patient table

Doctor

Doc-id	Primary Key
DName	
Qualification	
Salary	
Hosp-id	Foreign key references to Hosp-id of Hospital table

Step 3: Mapping of Attributes

- **Simple Attributes**
Simple Attributes which can not be divided into subparts.
Example: Salary of Doctor



- **Composite Attributes**

Composite Attributes which can be divided into subparts.

Example: Patient Name, Doctor Name

Patient

First_Name
Middle_Name
Last_name

Doctor

First_Name
Middle_Name
Last_name

Step 4: Mapping of Relationships

b. Foreign Key approach

Hosp_patient

Pat-id	Hospital table makes foreign key references to Pat-id of Patient table
Hosp-id	Patient table makes foreign key references to Hosp-id of Hospital table

Hosp_Doctor

Hosp-id	Doctor table makes foreign key references to Hosp-id of Hospital table
Doc-id	Hospital table makes foreign key references to Doc-id of Doctor table

PatiPatient_MedicalRecord

Pat-id	Medical Record table makes foreign key references to Pat-id of Patient table
Record-id	Patient table makes foreign key references to Record-id of Medical Record table

Step 5: Identifying the relationships

a. Hospital has a set of patients.

Therefore the relationship is 1.....N.

b. Hospital has a set of doctors.

Therefore the relationship is 1.....N.

c. Doctors are associated with each patient.

Therefore the relationship is N.....1.

d. Each patient has a record of various tests and examinations conducted.

Therefore the relationship is 1.....N.

Conclusion:

Outcome of the experiment is students are able to,

Design ER model and convert ER diagram into database tables for handling database as per given requirements

FAQS:

1. Explain the distinctions among the terms primary key, candidate key, and superkey
2. What is an entity? Give examples of entities
3. Define and give examples to illustrate the four types of attributes in the database.
4. Explain the four types of mapping cardinality with examples.
5. Define discriminator or partial key of a weak entity set. Give an example.

GROUP: A**ASSIGNMENT NO: 2A**

Aim: Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym

Problem Statement:

1. Create table Customers with schema (cust_id, cust_name, product, quantity, total_price)
2. Use sequence/ auto-increment for incrementing customer ID and Insert 5 customer records to the table Customers
3. Alter the table Customers by adding one column 'price_per_qnty'
4. Create view 'Cust_View' on Customers displaying customer ID, customer name
5. Update the view 'Cust_View' to display customer ID, product, total price
6. Drop the view 'Cust_View'
7. Create index 'Cust_index' on customer name
8. Drop index 'Cust_index'
9. Use sequence/ auto-increment for incrementing customer ID
10. Use the name alias for table Customers (rename the table in query)
11. Drop the table Customers

PREREQUISITE: Basic queries for creation of view, indexes & tables

Course Objectives:

- a. To understand the fundamental concepts of Database Management Systems
- b. To acquire the knowledge of database query languages and transaction processing
- c. To understand systematic database design approaches

Course Outcome:

1. Design E-R Model for given requirements and convert the same into database tables
2. Design schema in appropriate normal form considering actual requirements
3. Implement SQL queries for given requirements, using different SQL concepts

Theory:Creating Databases:The 'Create Database' statement is used to create a new SQL database. Database names should be unique within the RDBMS.

Syntax: create database <Database Name>;

Example: mysql> create database testDB;

The 'use' statement is used to select any existing database in SQL schema.

Syntax: use <Database Name>;

Creating Tables:Creating a basic table involves naming the table and defining its columns and each column's data type. The 'CREATE TABLE' statement is used to create a new table.

Syntax:

CREATE TABLE <table name>(<column1> <datatype>, <column2> <datatype>, <columnn> <datatype>, <column> <datatype>, PRIMARY KEY (one or more columns));

To describe table design, we can use "desc" query.

Department of Computer Engineering, MMCOE

Eg. mysql> desc table_name

Dropping Tables:

Syntax:

DROP TABLE table_name;

Creating Views:

Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables, or another view. To create a view, a user must have the appropriate system privilege according to the specific implementation.

Syntax

CREATE VIEW view_name AS SELECT column1, column2..... FROM table_name
WHERE [condition];

Dropping View:

Syntax

DROP VIEW view_name;

Creating Indexes:

Indexes are special lookup tables that the database search engine can use to speed up data retrieval. An index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

For example, if you want to reference all pages in a book that discuss a certain topic, you first refer to the index, which lists all topics alphabetically and are then referred to one or more specific page numbers. An index helps speed up SELECT queries and WHERE clauses, but it slows down data input, with UPDATE and INSERT statements. Indexes can be created or dropped with no effect on the data. Creating an index involves the CREATE INDEX statement, which allows you to name the index, to specify the table and which column or columns to index, and to indicate whether the index is in ascending or descending order. Indexes can also be unique, similar to the UNIQUE constraint, in that the index prevents duplicate entries in the column or combination of columns on which there's an index.

Syntax:

CREATE INDEX index_name ON table_name;

Dropping Index:

Syntax: DROP INDEX index_name;

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Auto-increment	Auto incrementing the Customer ID in customer table	Customer Ids are incrementing automatically	Pass
T_P2	Altering table	Writing SQL query for adding one new column to existing table Customer	'Price_per_Qnty' column added successfully	Pass
T_P3	Duplicate Primary Key	Inserting record with same primary key value to those of existing record in Customers table	Record doesn't get inserted	Fail

Conclusion:

Outcome of the experiment is students are able to

1. Create view, Index, synonyms and sequence.
2. Create schema with normalized form using various DDL statements

Department of Computer Engineering, MMCOE

FAQS:

Q.1. How is data independence of application programs ensured in a DBMS?

Q.2. List the different types of indexes?

Q.3. What is schema and instance?

Q4. What is the referential integrity key?

Q.5. When should indexes be avoided?

OUTPUT: Soft Copy of the program with output.

GROUP: A**ASSIGNMENT NO: 2B**

Aim : Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator

Problem Statement:

1. Create table Student with schema (roll_no, name, division, branch, city, marks)
2. Insert 10 records to the table students
3. List all the student names with their corresponding city
4. List all the distinct names of the students
5. List all the records of the students with all the attributes
6. List all the students whose marks are greater than 75
7. List all the students whose name starts with the alphabet 'S'
8. List all the students whose marks are in the range of 50 to 60
9. List all the students whose branch is 'computer' and city is 'Pune'
10. Update the branch of a student to 'IT' whose roll number is 9
11. Delete the student records whose division is 'BE'
12. Create another table TE_Students with Schema(roll_no, name)
13. List all the roll numbers unionly in the relations Student and TE_Students
14. Display name of all the students belonging to relation Student in Upper case
15. Display the binary and hex equivalent of marks for all the students belonging to Student relation

PREREQUISITE: Knowledge of Basic queries for SQL

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To understand and execute process of software application development

Course Outcome:

1. Design Schema with appropriate normal form and Implement SQL Queries for given requirements, Using different SQL Concepts

Theory:**Inserting records to the Relation:**

The INSERT INTO Statement is used to add new rows of data to a table in the database.

Syntax:

1. INSERT INTO TABLE_NAME (column1, column2, column3,...columnN) VALUES (value1, value2, value3,...valueN);

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

2. You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

3. Insert statements that use VALUES syntax can insert multiple rows. To do this, include multiple lists of column values, each enclosed within parentheses and separated by commas.

When you have to insert multiple records at the same time you can use like the following example *E.g.*

INSERT INTO tbl_name (a,b,c) VALUES(1,2,3),(4,5,6),(7,8,9);

where a,b,c are the columns of the table and (1,2,3), (4,5,6),(7,8,9) are the three records with the corresponding values.

Select:

The most commonly used SQL command is **SELECT statement**. It is used to query the database and retrieve selected data that follow the conditions we want. In simple words, we can say that the select statement used to query or retrieve data from a table in the database.

Syntax:

SELECT column1, column2, columnN FROM table_name;

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

SELECT * FROM table_name;

Clauses in Select Statement:

[WHERE Clause] : It specifies which rows to retrieve.

[GROUP BY Clause] : Groups rows that share a property so that the aggregate function can be applied to each group.

[HAVING Clause] : It selects among the groups defined by the GROUP BY clause.

[ORDER BY Clause] : It specifies an order in which to return the rows.

a) Select with Distinct Keyword:

The SQL DISTINCT command is used with SELECT key word to retrieve only distinct or unique data.

Syntax: SELECT DISTINCT column1, column2,...columnN FROM table_name WHERE [condition]

b) Select with AND/OR Operators

The SQL AND & OR operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called as the conjunctive operators. These operators provide a means to make multiple comparisons with different operators in the same SQL statement. The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause. You can combine N number of conditions using the AND operator. For an action to be taken by the SQL statement, whether it be a transaction or a query, all conditions separated by the AND must be TRUE.

Syntax: SELECT column1, column2, columnN FROM table_name WHERE [condition1] AND [condition2]...AND [conditionN];

The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause. You can combine N number of conditions using the OR operator. For an action to be taken by the SQL statement, whether it be a transaction or query, the only any ONE of the conditions separated by the OR must be TRUE.

Syntax: SELECT column1, column2, columnN FROM table_name WHERE [condition1] OR [condition2]...OR [conditionN]

c) Select with Like clause:

The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

The percent sign (%)

The underscore (_)

The percent sign represents zero, one or multiple characters. The underscore represents a single number or character. These symbols can be used in combinations.

d) Select with Limit clause

Eg: Select * from table_name limit 3; will generate the result set of first three records from the table

Update:

The SQL UPDATE Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

Syntax: UPDATE table_name SET column1 = value1, column2 = value2..., columnN = valueN WHERE [condition];

Delete:

The SQL DELETE Query is used to delete the existing records from a table. You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted. You can combine N number of conditions using AND or OR operators.

Syntax: DELETE FROM table_name WHERE [condition];

If you want to DELETE all the records from the CUSTOMERS table, you do not need to use the WHERE clause-

DELETE FROM table_name;

Refer the link <https://dev.mysql.com/doc/refman/5.7/en/func-op-summary-ref.html> for the list of functions available for mysql.

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Finding Binary equivalents	Display binary equivalents of roll numbers of students	By using BIN() function binary equivalents for roll numbers are displayed	Pass
T_P2	Inserting multiple records to same table	Writing SQL query for adding multiple records in student table in single query <<insert into student(columns for table) values (record 1), (record 2)>>	All the records are inserted successfully to table student	Pass
T_P3	Updating particular records	Update branch for student table without using where clause	All the records are updated for branch value instead of particular record	Fail

Conclusion:

Department of Computer Engineering, MMCOE

Outcome of the experiment is students are able to,

1. Execute various DML commands with its clauses, functions, operators and set operators.
2. Implement DML SQL Queries for the given requirements.

FAQS:

Q.1. List various types of constraints. Is a NULL value the same as zero or a blank space? If not then what is the difference?

Q.2. How do we use the DISTINCT statement? What is its use?

Q.3. What is the purpose of the condition operators BETWEEN and IN?

Q4. How do you search for a value in a database table when you don't have the exact value to search for?

Q 5. What is the default ordering of data using the ORDER BY clause? How could it be changed?

Q. 6. What happens if you omit the WHERE clause in the DELETE statement?

OUTPUT: Soft Copy of the program with output.

GROUP: A**ASSIGNMENT NO: 3**

Aim : Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query .

Problem Statement:

1. Create table Customers with schema (ID, name, age, address, salary)
2. Create table Orders with Schema(O_ID, o_date, customer_id, amount)
3. Insert 5 records to each table keeping few customer ids common to both the tables
4. Perform the inner join on customers and orders table to enlist the id, name, amount and o_date
5. Perform the left outer join on customers and orders table to enlist the id, name, amount and o_date
6. Perform the right outer join on customers and orders table to enlist the id, name, amount and o_date
7. Perform the full outer join on customers and orders table to enlist the id, name, amount and o_date by using 'union all' set operation
8. Perform the self join on customers table to enlist the pair of customers belonging to same address
9. Perform the Cross/ Cartesian join on customers and orders table to enlist the id, name, amount and o_date
10. Design the sub query with select statement for displaying all the details of the customers having salary greater than 20000
11. Create a backup table- 'cust_bkp' of the table customers by using insert statement with the subquery
12. Update the salaries by 10% of all the customers(in customers table) having age greater than or equals to 24 by using sub query with update clause(by using backup table cust_bkp)
13. Delete all the customers having age greater than 26 by using delete clause with the subquery

PREREQUISITE: Knowledge of all the join types supported by SQL.

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To understand and execute process of software application development

Course Outcome:

1. Design Schema with appropriate normal form and Implement SQL Queries for given requirements, Using different SQL Concepts

Theory:

Joins: The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal to symbol. There are different types of joins available in SQL

INNER JOIN – returns rows when there is a match in both tables.

LEFT JOIN – returns all rows from the left table, even if there are no matches in the right table.

RIGHT JOIN – returns all rows from the right table, even if there are no matches in the left table.

FULL JOIN – returns rows when there is a match in one of the tables.

SELF JOIN – is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

CARTESIAN JOIN – returns the Cartesian product of the sets of records from the two or more joined tables.

a) Inner Join: The most important and frequently used of the joins is the INNER JOIN. They are also referred to as an EQUIJOIN. The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

Syntax:

```
SELECT table1.column1, table2.column2... FROM table1 INNER JOIN table2  
ON table1.common_field = table2.common_field;
```

b) Left Join

The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in the right table; the join will still return a row in the result, but with NULL in each column from the right table.

This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.

Syntax:

```
SELECT table1.column1, table2.column2... FROM table1 LEFT JOIN table2  
ON table1.common_field = table2.common_field;
```

c) Right Join:

The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in the left table; the join will still return a row in the result, but with NULL in each column from the left table.

This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

Syntax:

```
SELECT table1.column1, table2.column2... FROM table1 RIGHT JOIN table2  
ON table1.common_field = table2.common_field;
```

d) Full Join:

The SQL FULL JOIN combines the results of both left and right outer joins. The joined table will contain all records from both the tables and fill in NULLs for missing matches on either side. MySQL does not support full join so we have to use the set operation 'UnionAll' for left outer join and the right outer join on those two tables.

Syntax:

```
<Left Outer Join Query> Union All <Right Outer Join Query>
```

e) Self Join:

The SQL SELF JOIN is used to join a table to itself as if the table were two tables; temporarily renaming at least one table in the SQL statement. Here, the WHERE clause could be any given expression based on your requirement.

Syntax: SELECT a.column_name, b.column_name... FROM table1 a, table1 b
WHERE a.common_field = b.common_field;

f) Cartesian or Cross Join:

The CARTESIAN JOIN or CROSS JOIN returns the Cartesian product of the sets of records from two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to either True or where the join-condition is absent from the statement.

Syntax: SELECT table1.column1, table2.column2...FROM table1, table2 [, table3]

Sub-Query in SQL: A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

Department of Computer Engineering, MMCOE

There are a few rules that subqueries must follow –

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

a) Sub-Query with Select Statement:

Subqueries are most frequently used with the SELECT statement.

Syntax:

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

b) Sub-Query with Insert Statement:

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

Syntax: INSERT INTO table_name [(column1 [, column2])]

```
SELECT [ *|column1 [, column2 ]
FROM table1 [, table2 ] [ WHERE VALUE OPERATOR ]
```

c) Sub-Query with Update Statement:

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

Syntax: UPDATE table SET column_name = new_value [WHERE OPERATOR [VALUE] (SELECT COLUMN_NAME FROM TABLE_NAME) [WHERE)]

d) Sub-Query with Delete Statement:

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

Syntax: DELETE FROM TABLE_NAME [WHERE OPERATOR [VALUE] (SELECT COLUMN_NAME FROM TABLE_NAME) [WHERE)]

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Full Join using Union all	SQL query can be written as <<left join query>> union all <<right join query>>	Result set is equivalent to actual full join query	Pass

Department of Computer Engineering, MMCOE

T_P2	Getting NULL values in left outer join	The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table and put NULL values for those unmatched attributes	NULL values obtained for unmatched attributes in right relation in join	Pass
------	--	--	---	------

Conclusion:

Outcome of the experiment is students are able to

1. Implement all types of Joins
2. Implement Subqueries with Select, Insert, Update and Delete statement

FAQS:

- Q.1. What is the difference between cross joins ,natural joins and self join?
Q.2. What do you understand by a nested query? When is it used?
Q.3. Can you modify the rows in a table based on values from another table?
Q.4 How many tables may be included with a join?
A. One B. Two C. Three D. All of the mentioned
Q. 5 Mention various set membership operators

OUTPUT: Soft Copy of the program with output.

GROUP: A**ASSIGNMENT NO: 4**

Aim : To study unnamed PL/ SQL code blocks by using Control structure and Exception handling.

Problem Statement: 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 the condition of fine is true, then details will be stored into a fine table.

PREREQUISITE: Knowledge of Basic SQL DDL, DML commands.

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To understand and execute process of software application development

Course Outcomes:

Implement PL/SQL Block for given requirements,Using different PL SQL Concepts

Theory:**Introduction to PL/SQL:**

PL/ SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL. PL/ SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java.

Features of PL/SQL

1. PL/SQL is tightly integrated with SQL.
2. It offers extensive error checking.
3. It offers numerous data types.
4. It offers a variety of programming structures.
5. It supports structured programming through functions and procedures.
6. It supports object-oriented programming.
7. It supports the development of web applications and server pages.

A Simple PL/SQL Block:PL/SQL Block consists of three sections:The Declaration section (optional).

The Execution section (mandatory).

The Exception Handling (or Error) section (optional).

Declaration Section:The Declaration section of a PL/SQL Block starts with the reserved keyword DECLARE. This section is optional and is used to declare any placeholders like variables, constants, records and cursors, which are used to manipulate data in the execution section. Placeholders may be any of Variables, Constants and Records, which stores data temporarily. Cursors are also declared in this section.

Execution Section:

The Execution section of a PL/SQL Block starts with the reserved keyword BEGIN and ends with END. This is a mandatory section and is the section where the program logic is written to perform any task. The programmatic constructs like loops, conditional statements and SQL statements form the part of the execution section.

Exception Section:

The Exception section of a PL/SQL Block starts with the reserved keyword EXCEPTION. This section is optional. Any errors in the program can be handled in this section, so that the PL/SQL Blocks terminates gracefully. If the PL/SQL Block contains exceptions that cannot be handled, the Block terminates abruptly with errors.

The following table lists few of the exceptions –

Exception	Oracle Error	SQL CODE	Description
ACCESS_INTO_NULL	06530	-6530	It is raised when a null object is automatically assigned a value.
CASE_NOT_FOUND	06592	-6592	It is raised when none of the choices in the WHEN clause of a CASE statement is selected, and there is no ELSE clause.
COLLECTION_IS_NULL	06531	-6531	It is raised when a program attempts to apply collection methods other than EXISTS to an uninitialized nested table or varray, or the program attempts to assign values to the elements of an uninitialized nested table or varray.
INVALID_NUMBER	01722	-1722	It is raised when the conversion of a character string into a number fails because the string does not represent a valid number.
LOGIN_DENIED	01017	-1017	It is raised when a program attempts to log on to the database with an invalid username or password.
NO_DATA_FOUND	01403	+100	It is raised when a SELECT INTO statement returns no rows.
NOT_LOGGED_ON	01012	-1012	It is raised when a database call is issued without being connected to the database.
PROGRAM_ERROR	06501	-6501	It is raised when PL/SQL has an internal problem.
STORAGE_ERROR	06500	-6500	It is raised when PL/SQL ran out of memory or memory was corrupted.
TOO_MANY_ROWS	01422	-1422	It is raised when a SELECT INTO statement returns more than one row.
VALUE_ERROR	06502	-6502	It is raised when an arithmetic, conversion, truncation, or size constraint error occurs.
ZERO_DIVIDE	01476	1476	It is raised when an attempt

Every statement in the below three sections must end with a semicolon ; . PL/SQL blocks can be nested within other PL/SQL blocks. Comments can be used to document code.

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling>

END;

Control Structure

Decision-making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false. Following is the general form of a typical conditional (i.e., decision making) structure found in most of the programming languages –

A **loop** statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages

Sample PL/SQL Block:

DECLARE

x NUMBER := 100;

BEGIN

FOR i IN 1..10 LOOP

IF MOD(i,2) = 0 THEN -- i is even

INSERT INTO temp VALUES (i, x, 'i is even');

ELSE

INSERT INTO temp VALUES (i, x, 'i is odd');

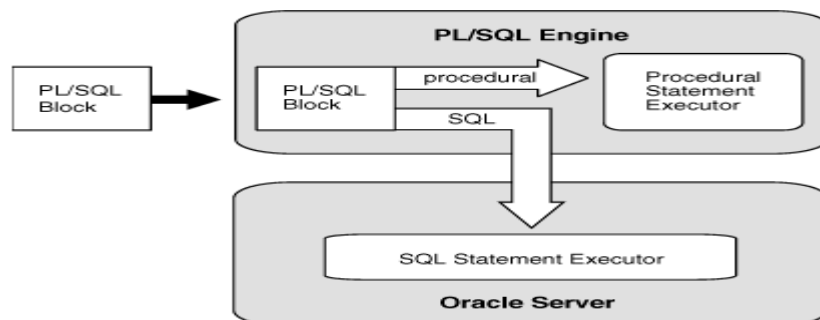
END IF;

x := x + 100;

END LOOP;

COMMIT;

END;/



Test Cases:

Department of Computer Engineering, MMCOE

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Calculate Fine	Fine of Rs.5 per day if days between 15 to 30	Total Fine displayed	Pass
T_P2	Exception	NO_DATA_FOUND	No rows selected	Pass
T_P3	Calculate Fine	Fine of Rs.5 per day if days <15	Fine to be displayed	Fail

Conclusion:

Outcome of the experiment is:

- Ability to execute PL/SQL blocks.
- Ability to write control structures, loop and exception handling

FAQS:

Q.1. Explain Importance of the PL/SQL language?

Q.2. List various schema objects that can be created using PL/SQL?

Q.3. Say True or False. Justify PL/SQL does not have data types or variables.

Q4. What is wrong in the following assignment statement?

`Acc_balance = Acc_balance * 0.20;`

Q 5. How to make use of %type data type? Explain with example.

Q.6. Define Exception Handling in PL/SQL? list any three.

OUTPUT: Soft Copy of the program with output.

GROUP: A**ASSIGNMENT NO: 5**

Aim : To study Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)

Problem Statement: 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_Roll-call. If the data in the first table already exists in the second table then that data should be skipped.

PREREQUISITE: Knowledge of PL/SQL block

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To understand and execute process of software application development

Course Outcomes:

Implement PL/SQL Block for given requirements, Using different PL SQL Concepts Cursor

Theory:

A cursor is a pointer to the context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set. You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors Implicit cursors, Explicit cursors

Implicit Cursors :

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it. Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement.

The following table provides the description of the most used attributes –

Sr. No	Attribute & Description
1	%FOUND :Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
2	%NOTFOUND: The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE
3	%ISOPEN: Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	%ROWCOUNT: Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Example of Implicit Cursor:

```
DECLARE
  total_rows number(2);
BEGIN
  UPDATE customers
  SET salary = salary + 500;
IF sql%notfound THEN
  dbms_output.put_line('no customers selected');
ELSE IF sql%found THEN
  total_rows := sql%rowcount;
  dbms_output.put_line( total_rows || ' customers selected ');
END IF;
END;/
```

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

Working with an explicit cursor includes the following steps –

1. Declaring the cursor for initializing the memory
2. Opening the cursor for allocating the memory
3. Fetching the cursor for retrieving the data
4. Closing the cursor to release the allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example –
CURSOR c_customers IS SELECT id, name, address FROM customers;

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the cursor as –

```
OPEN c_customers;
```

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, fetch rows from the above-opened cursor as follows –

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the opened cursor as –
CLOSE c_customers;

Example:

```
DECLARE
  c_id customers.id%type;
  c_name customerS.No.ame%type;
  CURSOR c_customers is
  SELECT id, name FROM customers;
BEGIN
```

```
OPEN c_customers;  
LOOP  
FETCH c_customers into c_id, c_name, c_addr;  
    EXIT WHEN c_customers%notfound;  
    dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);  
END LOOP;  
CLOSE c_customers;  
END;/
```

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Check Mismatch of Record	Compare Rollno !=RollNo	Record inserted and displayed	Pass
T_P2	Check matching of record	Check RollNo=RollNo	Tuple inserted in second table	Fail

Conclusion:

Outcome of experiment is students are able to

1. Implement implicit,explicit and parameterized cursors.
2. Execute cursors with various types of cursor attributes

FAQS:

Q.1. How To Execute the Cursor Queries with "OPEN" Statements?

Q.2. Explain use of cursor

Q.3. Write code of a cursor for loop.

Q.4. What is meant by % ROWTYPE and TYPE RECORD.

Q.5. Which is the default cursor in PL/SQL

OUTPUT: Printout of program with output.

GROUP: A**ASSIGNMENT NO: 6****Aim : PL/SQL Stored Procedure and Stored Function**

Problem Statement: Write a Stored Procedure namely proc_Grade for the categorization of students. If marks scored by students in examination is ≤ 1500 and marks ≥ 990 then students 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 procedures created with the above requirement. Stud_Marks(name, total_marks) Result(Roll, Name, Class)

PREREQUISITES:

1. Basics of PL/SQL block
2. Knowledge of Basic SQL DDL, DML commands.

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To understand and execute process of software application development

Course Outcomes:

Implement PL/SQL Block for given requirements, Using different PL SQL Concepts

Theory:

A subprogram is a program unit/module that performs a particular task. A subprogram can be created 1) At the schema level 2) Inside a package

Inside a PL/SQL block

PL/SQL provides two kinds of subprograms –

Functions– These subprograms return a single value; mainly used to compute and return a value.

Procedures– These subprograms do not return a value directly; mainly used to perform an action.

Creating a Procedure:

A procedure is created with the CREATE OR REPLACE PROCEDURE statement.

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
  < procedure_body >
END procedure_name;
```

Where, w

procedure-name specifies the name of the procedure.

[OR REPLACE] option allows the modification of an existing procedure.

The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.

Procedure-body contains the executable part.

The AS keyword is used instead of the IS keyword for creating a standalone procedure.

Executing a Standalone Procedure

A standalone procedure can be called in two ways –

- 1) Using the EXECUTE keyword
- 2) Calling the name of the procedure from a PL/SQL block

Deleting a Standalone Procedure

A standalone procedure is deleted with the DROP PROCEDURE statement. Syntax for deleting a procedure is –

DROP PROCEDURE procedure-name

The following table lists out the parameter modes in PL/SQL subprograms –

S.No	Parameter Mode & Description
1	IN : An IN parameter lets you pass a value to the subprogram. It is a read-only parameter. Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. You can pass a constant, literal, initialized variable, or expression as an IN parameter. You can also initialize it to a default value; however, in that case, it is omitted from the subprogram call. It is the default mode of parameter passing. Parameters are passed by reference.
2	OUT: An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. You can change its value and reference the value after assigning it. The actual parameter must be variable and it is passed by value.
3	IN OUT: An IN OUT parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and the value can be read. The actual parameter corresponding to an IN OUT formal parameter must be a variable, not a constant or an expression. Formal parameter must be assigned a value. Actual parameter is passed by value.

A function is same as a procedure except that it returns a value.

Creating a Function

A standalone function is created using the CREATE FUNCTION statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows –

```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
    < function_body >
END [function_name];
```

Where, function-name specifies the name of the function.

[OR REPLACE] option allows the modification of an existing function.

The function must contain a return statement.

The RETURN clause specifies the data type you are going to return from the function.

Function-body contains the executable part.

The AS keyword is used instead of the IS keyword for creating a standalone function.

Calling a Function

While creating a function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task. When a program calls a function, the program control is transferred to the called function. A called function performs the defined task and when its return statement is executed or when the last end statement is reached, it returns the program control back to the main program.

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Check Marks	Compare Range of marks	Record displayed	Pass
T_P2	Exception	If marks are below 600 and above 1500	Message displayed No Out of Bound	Fail

Conclusion:

Outcome of the experiment is students are able to

1. Implement stored procedure and function.
2. Execute drop procedure.

FAQS:

Q.1. What is the stored procedure?

Q.2. What is difference between FUNCTION and PROCEDURE in PL/SQL

Q.3. What are the advantages of stored procedure?

Q.4 Write when to use stored procedures to complete SQL Server tasks.

Q.5. What packages are available to PL SQL developers?

OUTPUT: Printout of program with output.

GROUP: A**ASSIGNMENT NO: 7**

Aim : To study Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).

Problem Statement: 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.

PREREQUISITES:

1. Basics of PL/SQL block
2. Knowledge of Basic SQL DDL, DML commands.

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To understand and execute process of software application development

Course Outcomes:

Implement PL/SQL Block for given requirements, Using different PL SQL Concepts Trigger

Theory:

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events –

A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)

A database definition (DDL) statement (CREATE, ALTER, or DROP).

A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers :

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Creating a trigger:

```
CREATE[ORREPLACE]TRIGGERtrigger_name
{BEFORE|AFTER|INSTEADOF} {INSERT[OR]||UPDATE[OR]||DELETE}
[OF col_name]ON table_name [REFERENCING OLD AS o NEWASn]
[FOR EACHROW] WHEN(condition)
```

DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

END;

Here,

Department of Computer Engineering, MMCOE

1. CREATE [OR REPLACE] TRIGGER trigger_name: It creates or replaces an existing trigger with the trigger_name.
2. {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
3. {INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
4. [OF col_name]: This specifies the column name that would be updated.
5. [ON table_name]: This specifies the name of the table associated with the trigger.
6. [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
7. [FOR EACH ROW]: This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
8. WHEN (condition): This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Following are the two very important point and should be noted carefully.

OLD and NEW references are used for record level triggers these are not available for table level triggers. If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Update record	Update record from library table	Record Updated	Pass
T_P2	Delete record	Delete record from table	Trigger not fired	Fail

Conclusion:

Outcome of the experiment is students are able to

1. Execute row level, statement level, Before and After Triggers.

FAQS:

Q. 1. How many types of triggers exist in PL/SQL?

Q.2. What is the difference between execution of triggers and stored procedures?

Q. 3. What happens when a trigger is associated with a view?

Q. 4. How would you reference column values BEFORE and AFTER you have inserted and deleted triggers?

Q . 5. How to disable a trigger name update_salary?

OUTPUT: Printout of program with output.

GROUP: A**ASSIGNMENT NO: 8**

Aim : Implement MYSQL database connectivity with Java. Implement Database navigation operations (add, delete, edit,) using JDBC.

Problem Statement:

1. Create table Stud with ID, name, marks columns using Java- JDBC connectivity
2. Insert 5 records in Stud
3. Update marks to 50 for student with ID =3;
4. Delete record for student whose ID is 1;
5. Alter table stud for adding one new column dept
5. Alter table stud and modify the column dept
6. Alter tables stud and drop column dept

PREREQUISITE: Knowledge of basic java connectivity steps with MYSQL.

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To understand and execute process of software application development

Course Outcome:

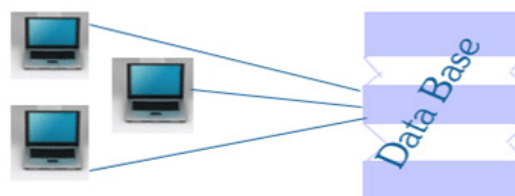
1. Implement connectivity with database using JDBC for given requirements

Theory:**Client -Data Server**

The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system.

The client–server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services. Servers are classified by the services they provide. For instance, a web server serves web pages and a file server serves computer files.

The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server.



The above figure shows the architecture of two-tier. Here the communication is one to one. For example, now we have a need to save the employee details in database. The two tiers of two-tier architecture is Database (Data tier), Client Application (Client tier). So, in client application the client writes the program

Department of Computer Engineering, MMCOE

for saving the record in SQL Server and thereby saving the data in the database.

JDBC:

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC library includes APIs for each of the tasks commonly associated with database usage:

Making a connection to a database, Creating SQL or MySQL statements, Executing that SQL or MySQL queries in the database, Viewing & Modifying the resulting records.

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as: Java Applications, Java Applets, Java Servlets, Java Server Pages (JSPs), Enterprise JavaBeans (EJBs). All of these different executables are able to use a JDBC driver to access a database and take advantage of the stored data. JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

What is JDBC Driver ?

JDBC drivers implement the defined interfaces in the JDBC API for interacting with your database server. For example, using JDBC drivers enable you to open database connections and to interact with it by sending SQL or database commands then receiving results with Java.

There are following six steps involved in building a JDBC application –

Import the packages: Requires that you include the packages containing the JDBC classes needed for database programming. Most often, using `import java.sql.*` will suffice.

`import java.sql.*;`

Register the JDBC driver: Requires that you initialize a driver so you can open a communication channel with the database.

`Class.forName("com.mysql.jdbc.Driver");`

Open a connection: Requires using the `DriverManager.getConnection()` method to create a connection object, which represents a physical connection with the database.

`connection`

`con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root","admin");`

Where root & admin are the credentials for DB and test is the required db to connect with.

Execute a query: Requires using an object of type `Statement` for building and submitting an SQL statement to the database.

`Statement st=con.createStatement();`

`String sql="<sql query>";`

`st.executeUpdate(sql) // or st.executeQuery(sql)`

if result set has to be generated. Extract data from result set:

`ResultSet rs= st.executeQuery(sql)`

`While (rs.next())`

`{`

`//statement for printing the records as per the query`

`}`

Clean up the environment: Requires explicitly closing all database resources versus relying on the JVM's garbage collection.

`con.close();`

Test Cases:

T_I D	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Connection	Check JDBC connectivity with MYSQL	Record Inserted	Pass
T_P2	Exception	Class not found	Connected with DB	Fail

Conclusion:

Outcome of the experiment is students are able to

Department of Computer Engineering, MMCOE

1. Make connectivity with MYSQL using JDBC.
1. Create a database with a front end.

FAQS:

Q.1. Explain the role of the client in 3 tier architecture?

Q.2. What is the role of the business layer in DBMS architecture?

Q.3. Elaborate data layer of DBMS architecture?

Q. 4. Differentiate between two tier and three tier architecture

Q.5. Summarize the concept of ODBC

OUTPUT: Soft Copy of the Program with output.

GROUP: B**ASSIGNMENT NO: 9**

Aim : Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)

Problem Statement:

1. Create Collection Employee
2. Insert 5 documents to Employee
3. Read all the documents
4. Display all the documents in a formatted manner
5. Insert another 3 documents with a single insert command
6. Insert one document using save() method instead of insert() method
7. Read all the employees whose name is 'Joe' and age is 25
8. Read all the employees whose salary is greater than 5000
9. Update the department of employee 'Joe' from 'Production' to 'Operations'
10. Increment the salary of 'Joe' by Rs. 2000
11. Add email Id for 'Joe' (using \$addToSet modifier with update)
12. Remove all the documents for the employees belonging to 'Operations' department
13. Sort all the documents according to the name of employees
14. Display only first three documents
15. Display all the documents except first 4
16. Use all the administrative commands of MongoDB

PREREQUISITE: Knowledge of basic queries for CRUD operations in MongoDB

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To develop skills to handle NoSQL Database
4. To understand and execute process of software application development

Course Outcome:

1. Create and Design NoSQL queries for given requirements to handle databases of varying complexities

Theory:

Inserting and Saving Documents : save() method:

Updates an existing document or inserts a new document, depending on its document parameter.

Modifiers to be used with Update Operation:

Usually only certain portions of a document need to be updated. You can update specific fields in a document using atomic update modifiers. Update modifiers are special keys that can be used to specify complex update operations, such as altering, adding, or removing keys, and even manipulating arrays and embedded documents.

\$inc- To increment the value of a field

\$set- Sets the value of a field

\$unset- Unsets the value of a field

\$push, \$each, \$slice- All are array modifiers and helps to push and select each element from the array

\$addToSet- Adding key value pairs to the existing array

a) Suppose we were keeping website analytics in a collection and wanted to increment a counter each time someone visited a page. We can use update modifiers to do this increment atomically. Each URL and its number of page views is stored in a document that looks like this:

```
{
  "_id" : ObjectId("4b253b067525f35f94b60a31"),
  "url" : "www.example.com",
  "pageviews" : 52
}
```

Every time someone visits a page, we can find the page by its URL and use the "\$inc" modifier to increment the value of the "pageviews" key:

```
> db.analytics.update({"url" : "www.example.com"}, {"$inc" : {"pageviews" : 1}})
```

b) If the user wanted to store his favorite book in his profile, he could add it using "\$set" :

```
> db.users.update({"_id" : ObjectId("4b253b067525f35f94b60a31")}, {"$set" : {"favorite book" : "War and Peace"}})
```

c) "\$set" can even change the type of the key it modifies. For instance, if our fickle user decides that he actually likes quite a few books, he can change the value of the "favorite book" key into an array:

```
> db.users.update({"name" : "joe"}, {"$set" : {"favorite book" : ["Cat's Cradle", "Foundation Trilogy", "Ender's Game"]}})
```

d) If the user realizes that he actually doesn't like reading, he can remove the key altogether with "\$unset" >

```
db.users.update({"name" : "joe"}, {"$unset" : {"favorite book" : 1}})
```

e) You can also use "\$set" to reach in and change embedded documents:

```
Eg: > db.blog.posts.findOne()
```

```
{
  "_id" : ObjectId("4b253b067525f35f94b60a31"),
  "Aim" : "A Blog Post",
  "content" : "...",
  "author" : {
    "name" : "joe",
    "email" : "joe@example.com"
  }
}
```

```
> db.blog.posts.update({"author.name" : "joe"}, {"$set" : {"author.name" : "joe schmoe"}})
```

f) Updating multiple documents:

To update multiple documents, set the multi option to true. For example, the following operation updates all documents where stock is less than or equal to 10:

```
E.g. db.books.update({ stock: { $lte: 10 } }, { $set: { reorder: true } }, { multi: true })
```

Logical Operations with Queries

The find method is used to perform queries in MongoDB. Querying returns a subset of documents in a collection, from no documents at all to the entire collection. Which documents get returned is determined by the first argument to find , which is a document specifying the query criteria. When we start adding key/value pairs to the query document, we begin restricting our search. This works in a straightforward way for most types: numbers match numbers, booleans match booleans, and strings match strings. Querying for a simple type is as specifying the value that you are looking for. For example, to find all documents where the value for "age" is 27, we can add that key/value pair to the query document:

```
> db.users.find({"age" : 27})
```

"\$lt" , "\$lte" , "\$gt" , and "\$gte" are all comparison operators, corresponding to <, <=, >, and >=, respectively. They can be combined to look for a range of values. For example, to look for users who are between the ages of 18 and 30, we can do this:

```
> db.users.find({"age" : {"$gte" : 18, "$lte" : 30}})
```

This would find all documents where the " age " field was greater than or equal to 18 AND less than or equal to 30.

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Update Document	Writing a Query to update document	Document Updated	Pass
T_P2	Logical operators	Writing a query to use logical operator \$and	Returns all documents that match the conditions of both clauses.	Pass
T_P3	Save Method	Writing collection name for which to save apply	Record doesn't get updated	Fail

Conclusion:

Outcome of the experiment is students are able to

1. Implement CRUD operations.
2. Implement Logical Operations with Queries

FAQS:

Q.1. Justify with proper query,can we update multiple documents at once

Q.2. Explain Upsert?

Q.3. When are the batch inserts useful?

Q.4. What are the different administrative commands of MongoDB?

Q.5. What does mongoimport do?

OUTPUT: Soft Copy of the Queries with output.

GROUP: B**ASSIGNMENT NO: 10**

Aim : Implement aggregation and indexing with suitable examples using MongoDB.

Problem Statement:

1. Create Collection Product
2. Insert the documents by considering the keys name, company, cost
3. Aggregate the documents in the collection by grouping company name and displaying minimum and maximum price of product for the same company
4. Show the sorted result on the basis of company
5. Display number of documents in the collection
6. Display distinct company names in the collection
7. Display company name with its count for documents
8. Limit the result for one document only
9. Limit the result by skipping first two documents
10. Insert 10 documents by using for loop in the collection staff by considering keys Staff_id, staff_name, age
11. Find the document where Staff_id is 2 and explain different parameters for running the query. Observe the number of scanned objects, time in milliseconds, type of cursor etc.
12. create index on staff_name
13. Run the query for point no 11 and observe the change in values for different parameters of the query
14. Find the document of the staff where age is 40 and staff_name is 'Karan'. Observe the index name used for running the query
15. create the index on age and Staff_name
16. Run the query for point no. 14 and observe the index used for running the querying
17. Drop the index created on age and Staff_name and run the query for point no. 14 again and observe the type of index
18. Drop the index created on Staff_name and run the query for pint no. 11 and observe the type of index
19. Create a unique index on Staff_name and try inserting documents with the duplicate Staff_names. Observe the result

PREREQUISITE: Knowledge of basic queries for aggregation and indexing

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To develop skills to handle NoSQL Database
4. To understand and execute process of software application development

Course Outcome: Create and Design NoSQL **aggregation and indexing** for given requirements to handle databases of varying complexities

Theory:**Aggregation**

Aggregations are operations that process data records and return computed results. MongoDB provides a rich set of aggregation operations that examine and perform calculations on the data sets. Running data aggregation on the mongod instance simplifies application code and limits resource requirements. Like queries, aggregation operations in MongoDB use collections of documents as an input and return results in the form of one or more documents.

Aggregation Pipeline

The aggregation pipeline is a framework for data aggregation modeled on the concept of data processing

Department of Computer Engineering, MMCOE

pipelines. Documents enter a multi-stage pipeline that transforms the documents into aggregated results. The MongoDB aggregation pipeline starts with the documents of a collection and streams the documents from one pipeline operator to the next to process the documents. Each operator in the pipeline transforms the documents as they pass through the pipeline. Pipeline operators do not need to produce one output document for every input document. Operators may generate new documents or filter out documents. Pipeline operators can be repeated in the pipeline. The `db.collection.aggregate()` method returns a cursor and can return result sets of any size. Pipeline operators appear in an array. Documents pass through the operators in a sequence.

1. \$project:

It reshapes a document stream. \$project can rename, add, or remove fields as well as create computed values and sub-documents.

```
> db.articles.aggregate({"$project" : {"author" : 1, "_id" : 0}})
```

By default, "_id" is always returned if it exists in the incoming document

2. \$match:

Filters the document stream, and only allows matching documents to pass into the next pipeline stage. \$match uses standard MongoDB queries. \$match filters documents so that you can run an aggregation on a subset of documents. For example, if you only want to find out stats about users in Oregon, you might add a "\$match" expression such as `{ "$match" : { "state" : "OR" } }`. "\$match" can use all of the usual query operators ("`$gt`", "`$lt`", "`$in`", etc.). Generally, good practice is to put "\$match" expressions as early as possible in the pipeline. This has two benefits: it allows you to filter out unneeded documents quickly and the query can use indexes if it is run before any projections or groupings.

3. **\$limit:** Restricts the number of documents in an aggregation pipeline.

4. **\$skip:** Skips over a specified number of documents from the pipeline and returns the rest.

5. **\$unwind:** Takes an array of documents and returns them as a stream of documents. Unwinding turns each field of an array into a separate document.

6. \$group:

Groups documents together for the purpose of calculating aggregate values based on a collection of documents. The output of \$group depends on how you define groups. Begin by specifying an identifier (i.e. an `_id` field) for the group you're creating with this pipeline. For this `_id` field, you can specify various expressions, including a single field from the documents in the pipeline, a computed value from a previous stage, a document that consists of multiple fields, and other valid expressions, such as constant or sub document fields. You can use \$project operators in expressions for the `_id` field.

The following example of an `_id` field specifies a document that consists of multiple fields:

```
{ $group: { _id : { author: '$author', pageViews: '$pageViews', posted: '$posted' } } }
```

7. \$sort:

Takes all input documents and returns them in a stream of sorted documents. It is highly recommended that you do the sort at the beginning of the pipeline and have an index it can use. Otherwise, the sort may be slow and take a lot of memory. You can use both existing fields and projected fields in a sort.

Expression Operators:

Expression operators calculate values within the Pipeline Operators.

1. \$group operators: we have so many operators used for grouping. \$addToSet, min, \$max, \$first, \$last, \$avg, \$push, \$sum are few of them.

2. Boolean operators: \$and, \$or, \$not

3. Comparison Operators: \$cmp, \$eq, \$gt, \$gte, \$lt, \$lte, \$ne

4. Arithmetic operators: \$add, \$divide, \$mod, \$multiply, \$subtract

5. String operators: \$concat, \$strcasecmp, \$substr, \$toLowerCase, \$toUpperCase

6. Array operators: \$size

7. Conditional Expressions: \$cond, \$ifNull

8. Date

operators:

\$dayOfYear, \$dayOfMonth, \$dayOfWeek, \$year, \$month, \$week, \$hour, \$minute, \$second, \$millisecond

Pipeline operators & Indexes:

The \$match and \$sort pipeline operators can take advantage of an index when they occur at the beginning of the pipeline.

Department of Computer Engineering, MMCOE

If your aggregation operation requires only a subset of the data in a collection, use the \$match, \$limit, and \$skip stages to restrict the documents that enter at the beginning of the pipeline. When placed at the beginning of a pipeline, \$match operations use suitable indexes to scan only the matching documents in a collection. Placing a \$match pipeline stage followed by a \$sort stage at the start of the pipeline is logically equivalent to a single query with a sort and can use an index. When possible, place \$match operators at the beginning of the pipeline.

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Aggregation	Writing a query to display number of documents in collection.	By using count() number of documents in collection are displayed	Pass
T_P2	Indexing	Writing query for creating index on documents.	Index Created	Pass
T_P3	Duplicate Entry	Try to insert same record without creating unique index	Record inserted	Fail

Conclusion:

Outcome of the experiment is students are able to

1. Implement aggregation and indexing with MongoDB.
2. Implement pipeline operators.

FAQS:

Q.1. What is dropDups parameter used while creating index?

Q.2. Explain use of \$first and \$last?

Q.3. Make use of pipeline operators such as \$push & \$pop in aggregation operation with an example?

Q.4. Define an aggregation operation?

Q.5. What is the purpose of \$unwind?

OUTPUT: Soft Copy of the Queries with output.

GROUP: B**ASSIGNMENT NO: 11**

Aim : Implement Map Reduce operation with suitable example using MongoDB.

Problem Statement:

1. Create collection Staff with keys name, age and address
2. Apply Map Reduce operation over Staff collection to display the name of staff and sum of ages of the staff having same name.

PREREQUISITE: Knowledge of basic queries for creating functions for map & reduce

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To develop skills to handle NoSQL Database
4. To understand and execute process of software application development

Course Outcome: Create and Design NoSQL Map Reduce for given requirements to handle databases of varying complexities

Theory:

Map-Reduce: Map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results. For map-reduce operations, MongoDB provides the map Reduce database command. All map-reduce functions in MongoDB are JavaScript and run within the mongod process. Map-reduce operations take the documents of a single *collection* as the *input* and can perform any arbitrary sorting and limiting before beginning the map stage. Map Reduce can return the results of a map-reduce operation as a document, or may write the results to collections. The input and the output collections may be sharded. In MongoDB, map-reduce operations use custom JavaScript functions to *map*, or associate, values to a key. If a key has multiple values mapped to it, the operation *reduces* the values for the key to a single object. The use of custom JavaScript functions provides flexibility to map-reduce operations. For instance, when processing a document, the map function can create more than one key and value mapping or no mapping. Map-reduce operations can also use a custom JavaScript function to make final modifications to the results at the end of the map and reduce operation, such as perform additional calculations. Map Reduce is a powerful and flexible tool for aggregating data. It can solve some problems that are too complex to express using the aggregation framework's query language. Map Reduce uses JavaScript as its "query language" so it can express arbitrarily complex logic. However, this power comes at a price: Map Reduce tends to be fairly slow and should not be used for real-time data analysis. Map Reduce can be easily parallelized across multiple servers. It splits up a problem, sends chunks of it to different machines, and lets each machine solve its part of the problem. When all the machines are finished, they merge all the pieces of the solution back into a full solution. Map Reduce has a couple of steps. It starts with the map step, which maps an operation onto every document in a collection. That operation could be either "do nothing" or "emit these keys with X values." There is then an intermediary stage called the shuffle step: keys are grouped and lists of emitted values are created for each key. The reduce takes this list of values and reduces it to a single element. This element is returned to the shuffle step until each key has a list containing a single value: the result.

Map-Reduce Behaviour:

In MongoDB, the map-reduce operation can write results to a collection or return the results inline. If you write map-reduce output to a collection, you can perform subsequent map-reduce operations on the same input collection that merge replace, merge, or reduce new results with previous results. When returning the results of a map reduce operation inline, the result documents must be within the BSON Document Size limit, which is currently 16 megabytes. For additional information on limits and restrictions on map-reduce

Department of Computer Engineering, MMCOE

operations, see the mapReduce reference page. MongoDB supports map-reduce operations on sharded collections. Map-reduce operations can also output the results to a sharded collection.

Map Reduce Example:

In the mongo shell, the `db.collection.mapReduce()` method is a wrapper around the `mapReduce` command. The following examples use the `db.collection.mapReduce()` method:

Consider the following map-reduce operations on a collection `orders` that contains documents of the following prototype: {

```
_id: ObjectId("50a8240b927d5d8b5891743c"),
cust_id: "abc123",
ord_date: new Date("Oct 04, 2012"),
status: 'A',
price: 25,
items: [ { sku: "mmm", qty: 5, price: 2.5 },
          { sku: "nnn", qty: 5, price: 2.5 } ]
}
```

To Return the Total Price per customer:

Perform the map-reduce operation on the `orders` collection to group by the `cust_id`, and calculate the sum of the price for each `cust_id`:

1) Define the map function to process each input document

In the function, `this` refers to the document that the map-reduce operation is processing.

The function maps the price to the `cust_id` for each document and emits the `cust_id` and price pair.

```
var mapFunction1 = function() {
    emit(this.cust_id, this.price);
};
```

2) Define the corresponding reduce function with two arguments `keyCustId` and `valuesPrices`:

The `valuesPrices` is an array whose elements are the price values emitted by the map function and grouped by `keyCustId`.

The function reduces the `valuesPrice` array to the sum of its elements.

```
var reduceFunction1 = function(keyCustId, valuesPrices)
{
    return Array.sum(valuesPrices);
};
```

3) Perform the map-reduce on all documents in the `orders` collection using the `mapFunction1` map function and the `reduceFunction1` reduce function.

```
>db.orders.mapReduce(mapFunction1,reduceFunction1,{out: "map_reduce_example" })
```

This operation outputs the results to a collection named `map_reduce_example`. If the `map_reduce_example` collection already exists, the operation will replace the contents with the results of this map-reduce operation. We can also use 'finalize' to perform the aggregation on map reduced result.

Test Cases:

T_ID	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Map Reduce	Using Map Reduce operations for collection to display sum.	Sum Displayed	Pass
T_P2	Without emit	Perform Map Reduce without emitting values	Map Reduce can not be performed	Fail

Conclusion:

Outcome of the experiment is students are able to

1. Implement Map Reduce Operation with MongoDB.

FAQS:

Q.1. Justify, Are the map & reduce functions developed in JavaScript?

Q.2. Summarize that Map Reduce works upon aggregation principles of MongoDB?

Q.3. Explain Map Reduce?

Q.4. Mention the function used to perform the map reduce operation?

Q.5. Does MongoDB supports map-reduce operations on sharded collections?

OUTPUT: Soft Copy of the Queries with output.

GROUP: B**ASSIGNMENT NO: 12**

Aim : Write a program to implement MongoDB database connectivity with Java. Implement Database navigation operations (add, delete, edit etc.) using JDBC.

PROBLEM STATEMENT:Implement connectivity with MongoDB using any Java application.

PREREQUISITE:Knowledge of basic java connectivity steps with MongoDB.

Course Outcome: Implement MongoDB database connectivity with JDBC

Theory:

Before we start using MongoDB in our Java programs, we need to make sure that we have MongoDB JDBC Driver and Java set up on the machine. Now, let us check how to set up MongoDB JDBC driver. You need to download the jar from the path Download mongo.jar. Make sure to download latest release of it. You need to include the mongo.jar into your classpath.

Connect to database

To connect database, you need to specify database name, if database doesn't exist then mongodb creates it automatically. Code snippets to connect to database would be as follows:

```
import com.mongodb.*;

public class MongoDBJDBC {

    public static void main( String args[] ){

        try{

            // To connect to mongodb server

            MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

            // Now connect to your databases

            DB db = mongoClient.getDB( "test" );

            System.out.println("Connect to database successfully");

            boolean auth = db.authenticate(myUserName, myPassword);

            System.out.println("Authentication: "+auth);

        } catch(Exception e){

            System.err.println( e.getClass().getName() + ": " + e.getMessage() );

        }

    }

}
```

Now, compile and run above program to create our database test. You can change your path as per your requirement. We are assuming current version of JDBC driver mongo-2.10.1.jar is available in the current path. To create a collection, createCollection() method of com.mongodb.DB class is used. To get/select a collection from the database, getCollection() method of com.mongodb.DBCollection class is used. To insert

Department of Computer Engineering, MMCOE

a document into mongodb, insert() method of com.mongodb.DBCollection class is used. To select all documents from the collection, find() method of com.mongodb.DBCollection class is used. This method returns a cursor, so you need to iterate this cursor. To update document from the collection, update() method of com.mongodb.DBCollection class is used. To delete first document from the collection, you need to first select the documents using findOne() method and then remove method of com.mongodb.DBCollection class.

Test Cases:

T_I D	T_NAME	CONDITION TO TEST	ACTUAL RESULT	STATUS (PASS/FAIL)
T_P1	Connection	Check JDBC connectivity with MongoDB	Record Inserted	Pass
T_P2	Exception	Class not found	Connected with DB	Fail

Conclusion:

Outcome of the experiment is students are able to

1. Make connectivity with MongoDB using JDBC.
2. Create a database with a front end.

FAQS:

Q.1. Which method is used to delete the document?

Q.2. Which method is used to insert the document?

Q.3. Write port no of JDBC for making database connectivity of NoSQL

Q.4. List the alternatives to MongoDB.

Q.5. Mention what is Object_Id composed of?

OUTPUT: Soft Copy of the program with output.

GROUP: C**Assignment : 13**

Aim : To implement mini project

PROBLEM STATEMENT: Using the database concepts covered in SQL and NoSQL, develop an software application :

PREREQUISITES: Knowledge of basic queries of SQL/Oracle and NoSQL

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To develop skills to handle NoSQL Database
4. To understand and execute process of software application development

Course Outcome:

1. Design ER model and convert ER diagram into database tables
2. Design Schema with appropriate normal form and Implement SQL Queries for given requirements, Using different SQL Concepts
3. Implement PL/SQL Block for given requirements, Using different PL SQL Concepts
4. Create and Design NoSQL queries for given requirements to handle databases of varying complexities
5. Implement full database operation using software development lifecycle

Instructions:

1. Follow the same problem statement decided in Assignment -1 of Group A.
2. Follow the Software Development Life cycle and other concepts learnt in Software Engineering Course throughout the implementation.
3. Develop application considering:

Front End : Java/Perl/PHP/Python/Ruby/.net/any other language

Backend : MongoDB/MySQL/Oracle

4. Test and validate application using Manual/Automation testing.
5. Student should develop application in group of 2-3 students and submit the Project Report which will consist of documentation related to different phases of Software Development

Life Cycle: Report writing format

Title of the Project, Abstract, Introduction

Software Requirement Specification

Conceptual Design using ER features, Relational Model in appropriate Normalize form

Graphical User Interface, Source Code

Testing document

Conclusion. Outcome of the experiment is students are able to

Implementation of database system

ASSIGNMENT NO: 14**Content Beyond Syllabus**

Aim : Design star schema for Hospital management System using ERDPlus (Content Beyond Syllabus)

PROBLEM STATEMENT: Design star schema for Hospital management System using ERDPlus (Content Beyond Syllabus) use your mini project statement for drawing this scehmas

PREREQUISITES: Knowledge of database and SRS

Course Objectives:

1. To develop Database programming skills
2. To develop basic Database administration skills
3. To understand and execute process of software application development

Course Outcome:

1. Design ER model and convert ER diagram into database tables
2. Design Schema with appropriate tool to handle advanced database concepts

Theory:

A *data warehouse* is a type of **data management** system that is designed to enable and support business intelligence (BI) activities, especially analytics. Data warehouses are solely intended to perform queries and analysis and often contain large amounts of historical data. The data within a data warehouse is usually derived from a wide range of sources such as application log files and transaction applications.

A data warehouse centralizes and consolidates large amounts of data from multiple sources. Its analytical capabilities allow organizations to derive valuable business insights from their data to improve decision-making. Over time, it builds a historical record that can be invaluable to data scientists and business analysts. Because of these capabilities, a data warehouse can be considered an organization's "single source of truth."

A typical data warehouse often includes the following elements:

- A **relational database** to store and manage data
- An extraction, loading, and transformation (ELT) solution for preparing the data for analysis
- Statistical analysis, reporting, and data mining capabilities
- Client analysis tools for visualizing and presenting data to business users
- Other, more sophisticated analytical applications that generate actionable information by applying **data science** and artificial intelligence (AI) algorithms, or **graph** and spatial features that enable more kinds of analysis of data at scale

Benefits of a Data Warehouse

Data warehouses offer the overarching and unique benefit of allowing organizations to analyze large amounts of variant data and extract significant value from it, as well as to keep a historical record.

Four unique characteristics (described by computer scientist William Inmon, who is considered the father of the data warehouse) allow data warehouses to deliver this overarching benefit. According to this definition, data warehouses are

- **Subject-oriented.** They can analyze data about a particular subject or functional area (such as sales).
- **Integrated.** Data warehouses create consistency among different data types from disparate sources.
- **Nonvolatile.** Once data is in a data warehouse, it's stable and doesn't change.
- **Time-variant.** Data warehouse analysis looks at change over time.

A well-designed data warehouse will perform queries very quickly, deliver high data throughput, and provide enough flexibility for end users to “slice and dice” or reduce the volume of data for closer examination to meet a variety of demands—whether at a high level or at a very fine, detailed level. The data warehouse serves as the functional foundation for middleware BI environments that provide end users with reports, dashboards, and other interfaces.

Data Warehouse Architecture

The architecture of a data warehouse is determined by the organization's specific needs. Common architectures include

- **Simple.** All data warehouses share a basic design in which metadata, summary data, and raw data are stored within the central repository of the warehouse. The repository is fed by data sources on one end and accessed by end users for analysis, reporting, and mining on the other end.
- **Simple with a staging area.** Operational data must be cleaned and processed before being put in the warehouse. Although this can be done programmatically, many data warehouses add a staging area for data before it enters the warehouse, to simplify data preparation.
- **Hub and spoke.** Adding data marts between the central repository and end users allows an organization to customize its data warehouse to serve various lines of business. When the data is ready for use, it is moved to the appropriate data mart.
- **Sandboxes.** Sandboxes are private, secure, safe areas that allow companies to quickly and informally explore new datasets or ways of analyzing data without having to conform to or comply with the formal rules and protocol of the data warehouse.

Note: Refer SRS mentioned at assignment no-1 of Group A.

Schema is a logical description of the entire database. It includes the name and description of records of all record types including all associated data-items and aggregates. Much like a database, a data warehouse also requires to maintain a schema. A database uses relational model, while a data warehouse uses Star, Snowflake, and Fact Constellation schema. In this chapter, we will discuss the schemas used in a data warehouse.

Star Schema

Each dimension in a star schema is represented with only one-dimension table.

This dimension table contains the set of attributes.

The following diagram shows the sales data of a company with respect to the four dimensions, namely time, item, branch, and location

There is a fact table at the center. It contains the keys to each of four dimensions.

The fact table also contains the attributes, namely dollars sold and units sold.

Snowflake Schema

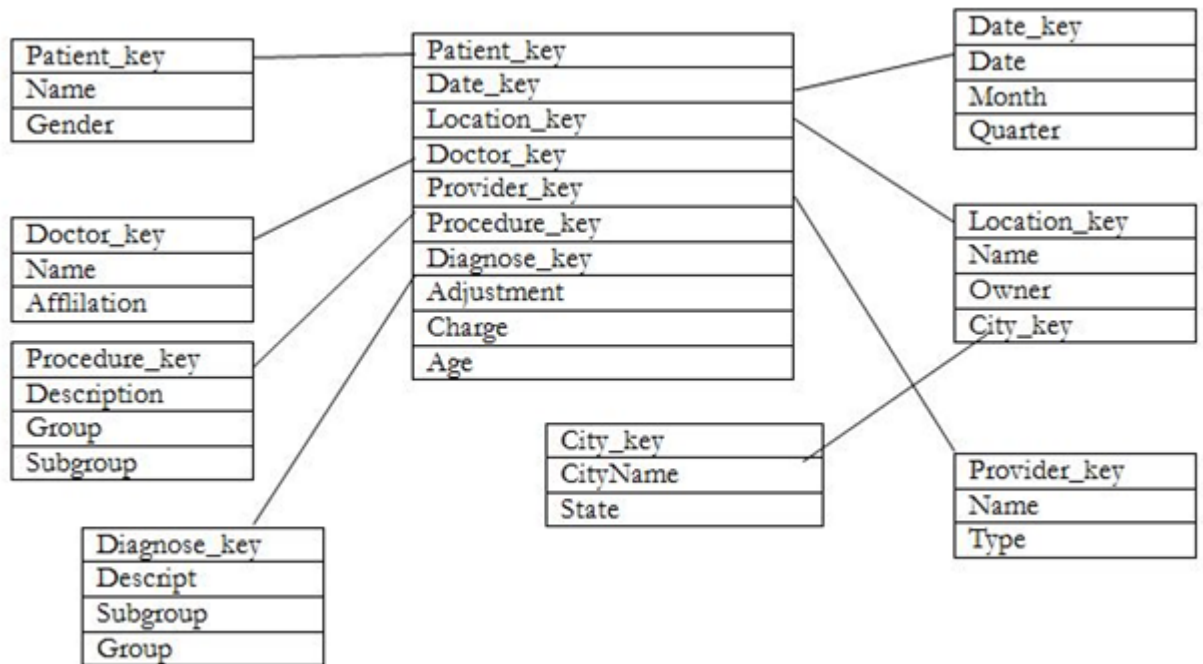
- Some dimension tables in the Snowflake schema are normalized.
- The normalization splits up the data into additional tables.
- Unlike Star schema, the dimensions table in a snowflake schema are normalized. For example, the item dimension table in star schema is normalized and split into two dimension tables, namely item and supplier table.

Dimensions:

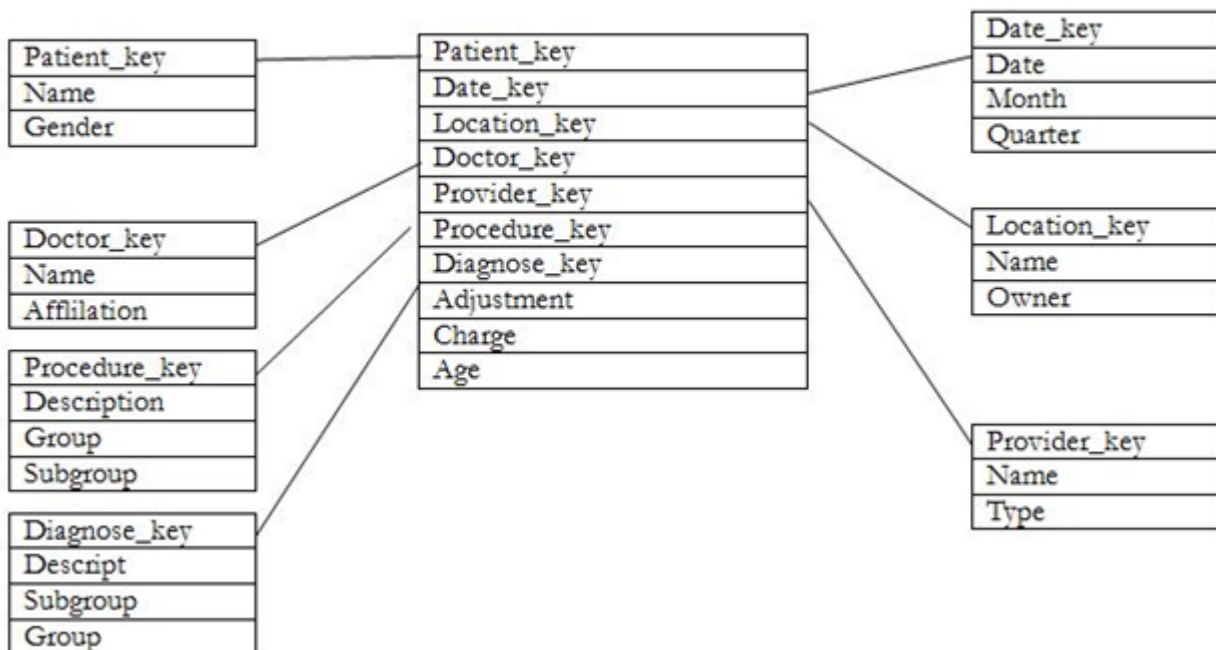
1. Patient
2. Doctor
3. Procedure
4. Diagnose
5. Date of Service
6. Location
7. Provider

Fact:

1. Adjustment
2. Charge
3. Age



Fact table for Procedure &
Billing History



Conclusion:

Outcome of the experiment is students are able to design star and snowflake schema.

OUTPUT: Printout of schemas .

Department of Computer Engineering, MMCOE

VLABS: Virtual Dev**ASSIGNMENT NO: 15**

Aim : Data Definition Language(DDL) Statements: (Create table, Alter table, Drop table)

Data Manipulation Language(DML) Statements

Data Query Language(DQL) Statements: (Select statement with operations like Where clause, Order by, Logical operators, Scalar functions and Aggregate functions)

Problem Statement: **To Understand and Implement Data Defining Language (DDL) Statements.**

Objectives:

- **Creating a table, with or without constraints.**
- Understanding Data types.
- Altering the structure of the table like adding attributes at later stage, modifying size of attributes or adding constraints to attributes.
- Removing the table created, i.e Drop table in SQL.

Outcomes: students are able to,

- **create table, drop table and alter table of DDL statements**
- **Implement Data Defining Language (DDL) Statements with Vlab**

Theory:

1] CREATE TABLE: The CREATE TABLE statement is used to create a table in SQL. A table comprises of rows and columns. So while creating tables we need to provide all the information to SQL about the names of the columns, type of data to be stored in columns, size of the data, constraints if any, etc.

i] Syntax:

create table <table_name>

(<column_name> <data_type> (<size>) [primary key] [not null], <column_name> <data_type> (<size>) [primary key] [not null], <column_name> <data_type> (<size>) [primary key] [not null], <column_name> <data_type> (<size>) [primary key] [not null], [primary key (<column_name>[, <column_name>, <column_name>..., <column_name>])]);

ii] Description:

table_name : name of the table.

column1 : name of the first column.

data_type : Type of data we want to store in the particular column.

For example number for numeric data.

size : Size of the data we can store in a particular column.

For example if for a column we specify the data_type as number and size as 10 then this column can store number of maximum 10 digits.

iii] Example

If you want to create a table called Employee which has all the attributes like Empid, Ename, Address, Salary.

Then, issue the following command in the editor Create table Employee (Empid varchar(6) , Ename varchar(15) , Address varchar(25) , Salary number(5));

2] ALTER TABLE

The SQL ALTER TABLE command is used to add, delete or modify columns in an existing table. You can also use the ALTER TABLE command to add and drop various constraints on an existing table.

Department of Computer Engineering, MMCOE

i] Syntax:a] Add a New Column The basic syntax of an ALTER TABLE command to add a New Column in an existing table is as follows.

```
alter table <table_name>
add ( <new_column_name> <new_data_type> (<new_size>) ,
<new_column_name> <new_data_type> (<new_size>) , .
<new_column_name> <new_data_type> (<new_size>) );
```

b] Modify column

It is used to modify the existing columns definition in a table. Multiple columns definitions can also be modified at once. The basic syntax of an ALTER TABLE command to change the DATA TYPE and SIZE of a column in a table is as follows:

```
alter table <table_name>
<modify> <column_name> <new_data_type> (<new_size>),
<column_name> <new_data_type> (<new_size>),
.
.
<column_name> <new_data_type> (<new_size>);
```

ADD PRIMARY KEY

The basic syntax of an ALTER TABLE command to ADD PRIMARY KEY constraint to a table is as follows.

```
alter table <table_name>
add primary key
(<column_name>[,<column_name>,<column_name>..,<column_name>]);
```

DROP PRIMARY KEY

```
alter table <table_name>
drop primary key;
```

c] Drop column

DROP COLUMN is used to drop column in a table. The user can use this command to delete the unwanted columns from the table. The basic syntax of an ALTER TABLE command to DROP COLUMN in an existing table is as follows.

```
alter table <table_name>
drop column
< column_name > [,<column_name>,<column_name>..,<column_name >];
```

3] DROP TABLE

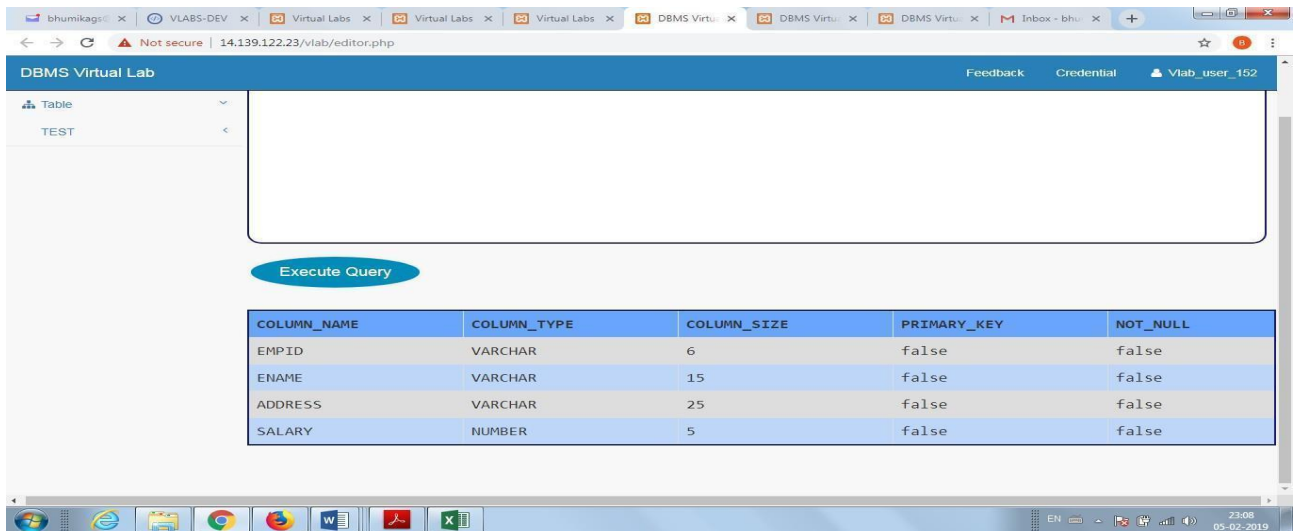
The SQL DROP TABLE statement is used to remove a table definition and all the data and constraints for that table. Once a table is deleted then all the information available in that table will also be lost forever.

i) Syntax:

drop table <table_name>;

ii) Example: Let us first verify the Employee table and then we will delete it from the database as shown below

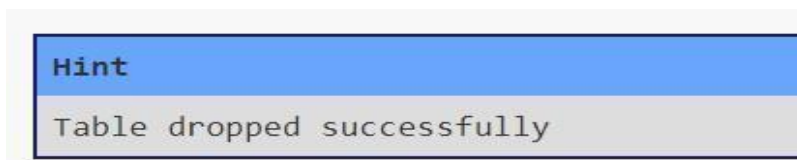
SQL> DESC Employee;



COLUMN_NAME	COLUMN_TYPE	COLUMN_SIZE	PRIMARY_KEY	NOT_NULL
EMPID	VARCHAR	6	false	false
ENAME	VARCHAR	15	false	false
ADDRESS	VARCHAR	25	false	false
SALARY	NUMBER	5	false	false

This means that the Employee table is available in the database, so let us now drop it as shown below.

SQL> DROP table Employee;



Now, if you would try the DESC command, then you will get the following error

SQL> DESC Employee;

"table doesn't exist : Employee"

Overview of Data Types

A **data type** is a classification of a particular type of information or data. Each value manipulated by database has a data type.

The data type of value associates a fixed set of properties with the value.

These properties allow the database to treat values of one data type differently from values of another.

Department of Computer Engineering, MMCOE

Data Types Supported by the Database*Character_ data types*

```
{  
VARCHAR (size [ BYTE | CHAR ])  
}
```

Number_ data types

```
{  
NUMBER [ (precision [, scale ]) ]  
}
```

Date_datatypes

```
{  
DATE(  
}
```

Data Type	Description
VARCHAR(size [BYTE CHAR])	Variable-length character string having maximum length size bytes or characters. You must specify size for VARCHAR. BYTE indicates that the column will have byte length semantics. CHAR indicates that the column will have character semantics.
NUMBER	Number is used for store the numerical values in database
DATE	This data type contains the date time fields YEAR, MONTH, DAY, HOUR, MINUTE and SECOND. It does not have fractional seconds or a time zone

Reference: <http://iitkgp.vlab.co.in/?sub=38&brch=204&sim=520&cnt=1>

Conclusion:

Outcome of the experiment is students are able to design ER diagram.

OUTPUT: Hardcopy of experiment with score and feedback.