Dr. D. Y.Patil Unitech Society

Dr. D. Y. Patil Institute of Technology

Sant Tukaram Nagar, Pimpri, Pune-411 018.

DEPARTMENT OF COMPUTER ENGINEERING

LABORATORY MANUAL

Third Year

310246: Database Management Systems Laboratory

(2019 Pattern)

Course Incharge

Savita Kumbhare

Rajesh Bharti

Jithina Jose

Aarju Jain

Academic Year 2023-24

SYLLABUS STRUCTURE

Savitribai Phule Pune University

Third Year of Computer Engineering (2019 Course)

310246: Database Management Systems Laboratory

Teaching Scheme Practical: 04 Hours/Week	**Credit 02**	**Examination Scheme and MarksTerm 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

Course Outcomes:

On completion of the course, learner 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 Laboratory /Term Work Assessment

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 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

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.

Virtual Laboratory: http://vlabs.iitb.ac.in/vlabs-dev/labs/dblab/labs/index.php

	Suggested List of Laboratory Experiments/Assignments(any 10)
Sr. No.	**Assignments**
	Group A: SQL and PL/SQL
1.	**ER Modeling and Normalization:** 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. 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.
2.	**SQL Queries:** a. Design and Develop SQLDDL statements which demonstrate the use of SQL objects suchas Table, View, Index, Sequence, Synonym, different constraints etc. b. Write at least 10 SQL queries on the suitable database application using SQL DML statements. **Note**: Instructor will design the queries which demonstrate the use of concepts like Insert, Select, Update, Delete with operators, functions, and set operator etc.
3.	**SQL Queries – all types of Join, Sub-Query and View:** Write at least10 SQL queries for suitable database application using SQL DML statements. **Note**: Instructor will design the queries which demonstrate the use of concepts like all types of Join ,Sub-Query and View
4.	Unnamed PL/SQLcode block: Use of Control structure and Exception handling is mandatory. Suggested Problem statement: Consider Tables: 1. Borrower(Roll_no, Name, Date of Issue, Name of Book, Status) 2. Fine(Roll_no, Date, Amt) • Accept Roll_no and Name of Book from user. • Check the number of days (from date of issue). • If days are between 15 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.
5.	• 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. **OR**

	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. **Note**: Instructor will frame the problem statement for writing PL/SQL block in line with above statement.
6.	Named PL/SQL Block: 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 and900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQLblock to use procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class) **Note**: Instructor will frame the problem statement for writing stored procedure and Function in line with above statement.
7.	**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_Roll Call 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. **Note**: Instructor will frame the problem statement for writing PL/SQL block using all types of Cursors in line with above statement.
8.	**Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).** Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table. **Note**: Instructor will Frame the problem statement for writing PL/SQLblock for all types of Triggers in line with above statement.
9.	**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.)
	Group B: NoSQL Databases
1.	**MongoDB Queries:** DesignandDevelopMongoDBQueriesusingCRUDoperations.(UseCRUDoperations, SAVE method, logical operators etc.).
2.	**MongoDB – Aggregation and Indexing**: Design and Develop MongoDB Queries using aggregation and indexing with suitable example using MongoDB.
3.	**MongoDB – Map-reduces operations:** Implement Map reduces operation with suitable example using MongoDB.
4.	**Database Connectivity:** Write a program to implement Mongo DB database connectivity with any front end language to implement Database navigation operations(add, delete, edit etc.)

	Group C: Mini Project
1.	Using the database concepts covered in Group A and Group B, develop an application with following details: 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: • 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. **Note**: • Instructor should maintain progress report of mini project through out 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

CO\ PO	**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**

INDEX

	suitable example using MongoDB.	
11.	**MongoDB - Map reduces operations:** Implement Map reduces operation with suitable example using MongoDB.	**75**
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.)	**78**
13.	Using the **database concepts covered in Group A and Group B**, develop an application with following details: 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: • 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.	**-**

ASSIGNMENT NO: 01

AIM:

ER Modeling and Normalization

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.

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.

OBJECTIVES:

- To develop Database programming skills

OUTCOMES :

- Design E-R Model for given requirements and convert the same into database tables.
- Design schema in appropriate normal form considering actual requirements

ENVIRONMENT:

Any tool for drawing ER diagram

THEORY:

Entity Relationship (E R) Model :

The Entity Relationship (ER) model is one of several high-level, or semantic, datamodels used in database design. The goal is to create a simple description of the data that closely matches how users and developers think of the data.

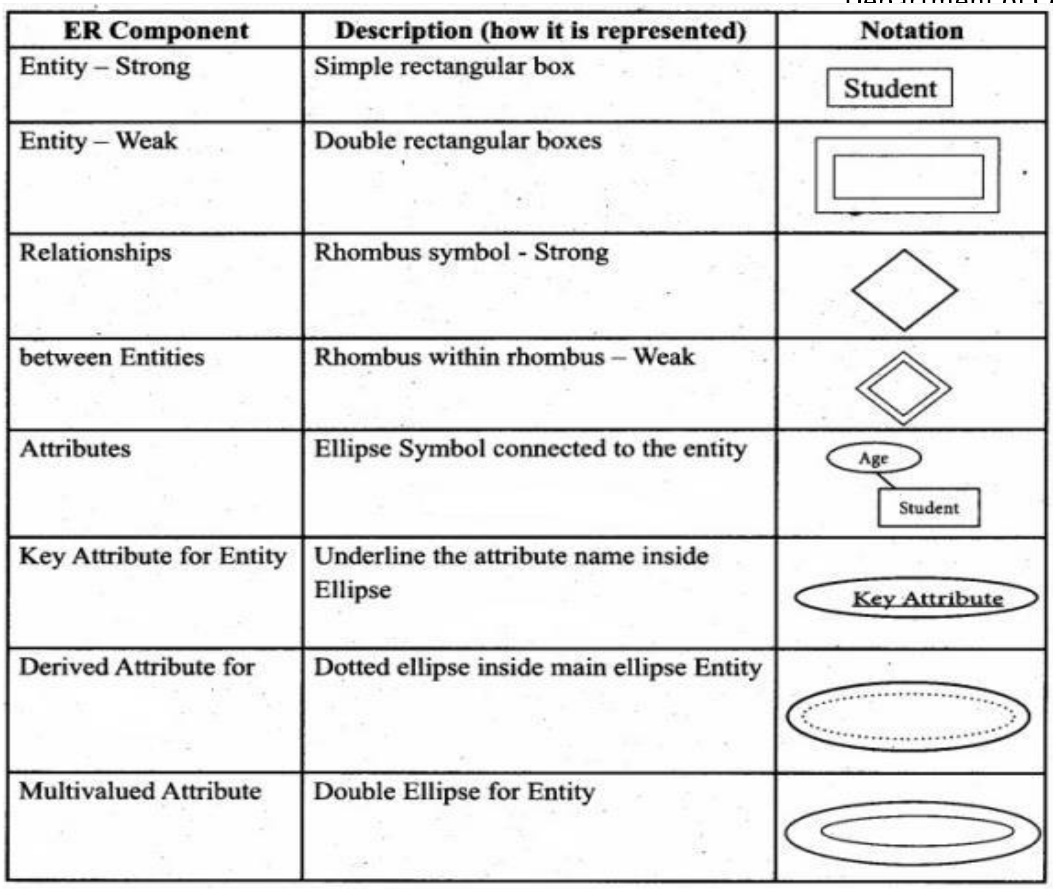
A database can be modeled as : a collection of entities, relationship among entities.An Entity is real-world object that exists and is distinguishable from other objects. A relationship is an association among several (Two or more) entities.

Entities are represented by means of their properties, called attributes. An entity set is a set of entities of the same type that share the same properties.Each entity set has a Key. Each Attribute has a Domain.

Types of Attributes

- **Simple attribute** − Simple attributes are atomic values, which cannot be divided further.

For example, a Customer's ID number is an atomic value of 6 digits.



ER Component	Description (how it is represented)	Notation
Entity – Strong	Simple rectangular box	Student
Entity – Weak	Double rectangular boxes	
Relationships	Rhombus symbol - Strong	
between Entities	Rhombus within rhombus – Weak	
Attributes	Ellipse Symbol connected to the entity	Age Student
Key Attribute for Entity	Underline the attribute name inside Ellipse	Key Attribute
Derived Attribute for	Dotted ellipse inside main ellipse Entity	
Multivalued Attribute	Double Ellipse for Entity	

- **Composite attribute** − Composite attributes are made of more than one simple attribute.

For example, a customer's complete name may have first-name, middle-initial and last-name.

Single-value attribute − Single-value attributes contain single value.

For example − Customer_ID, Social_Security_Number.

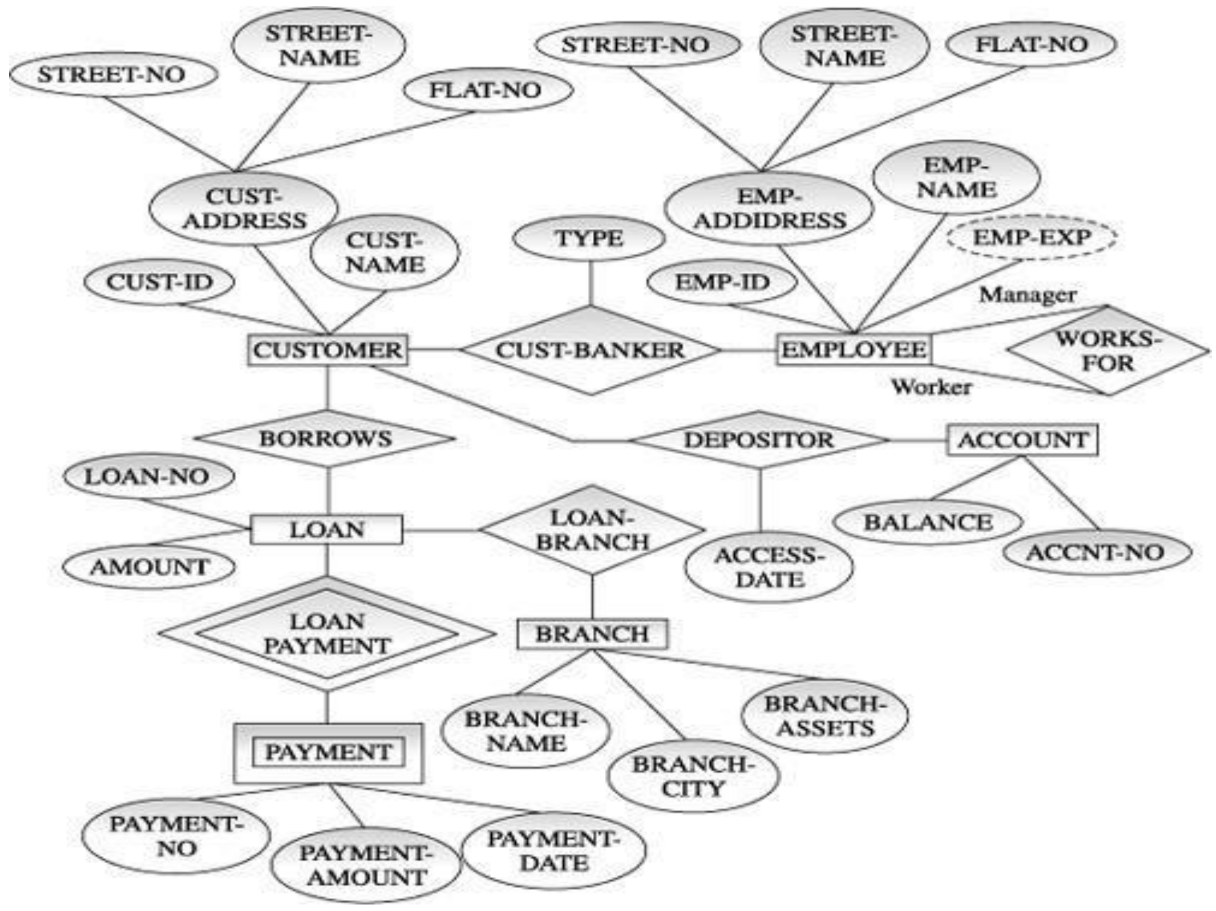
- **Multi-value attribute** − Multi-value attributes may contain more than one values.

For example, a person can have more than one phone number, mail_address, etc.

- **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but theirvalues are derived from other attributes present in the database.

For example, age can be derived from date_of_birth

Example of ER diagram



Relational Model

The relational model is a depiction of how each piece of stored information relates to the other stored information. It shows how tables are linked, what type of the links are between tables, what keys are used, what information is referenced between tables. It's an essential part of developing a normalized database structure to prevent repeat and redundant data storage. Different types of keys:

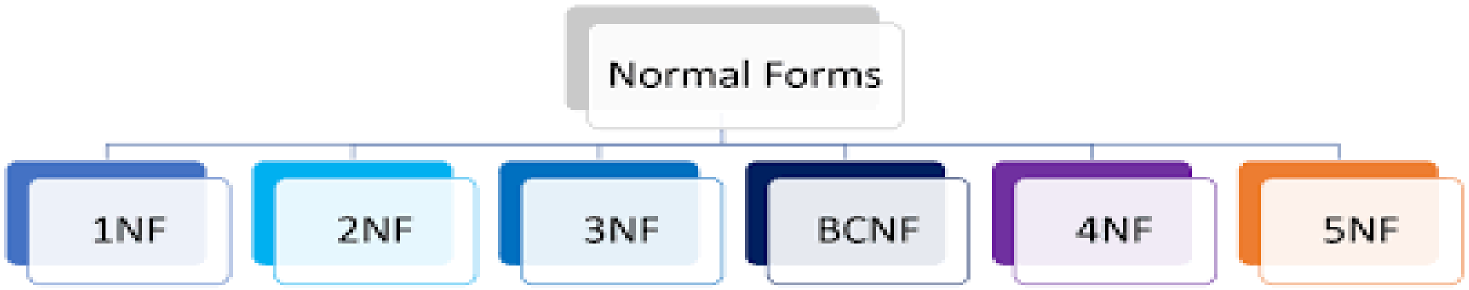
- A **super key** is a set of one or more attributes which; taken collectively, allow us to identify uniquely an entity in the entity set.
- A **primary key** is a candidate key(there may be more than one) chosen by the DB designer to identify entities in an entity set.
- A **super key** may contain extraneous attributes, and we are often interested in the smallest super key. A super key for which no subset is a super key is called a candidate key.
- An entity does not posses sufficient attributes to form a primary ket is called a **weak entity** set. One that does have a primary key is called a strong entity set.
- A **foreign key** is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.

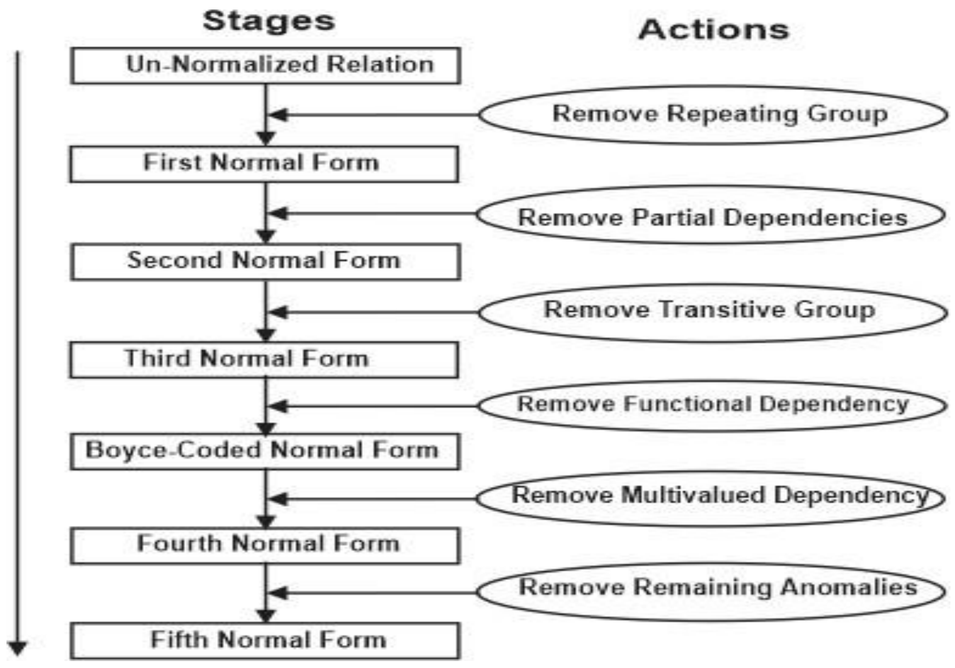
Normalization

Database normalization is a technique for designing relational database tables to minimize duplication of information and, in so doing, to safeguard the database against certain types of logical or structural problems, namely data anomalies. In thiswe will write the normalization tables that is entities of "Roadway Travels."

Normalization: In relational databases, normalization is a process that eliminates redundancy, organizes data efficiently; Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a tablet). Both of these are worthy goals as they reduce the amt of space a database consumes and ensure that data is logically stores.

The Normal Form: the database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one (the lowest form to normalization, referred to as first form or INF) through five(fifth normal form of SNF). In practical applications, you'll often see INF, 2NF, and 3NF along with occasional 4NF. Fifth normal form is very rarely seen and won't be discussed in this article. It's important to point out that they are guidelines and guidelines only. Occasionally, it becomes necessary to stray from them to meet practical business requirements. However, when variations take place, it's extremely important to evaluate any possible requirements they could have on your system and account for possible inconsistencies. That said, let's explore the normal form.

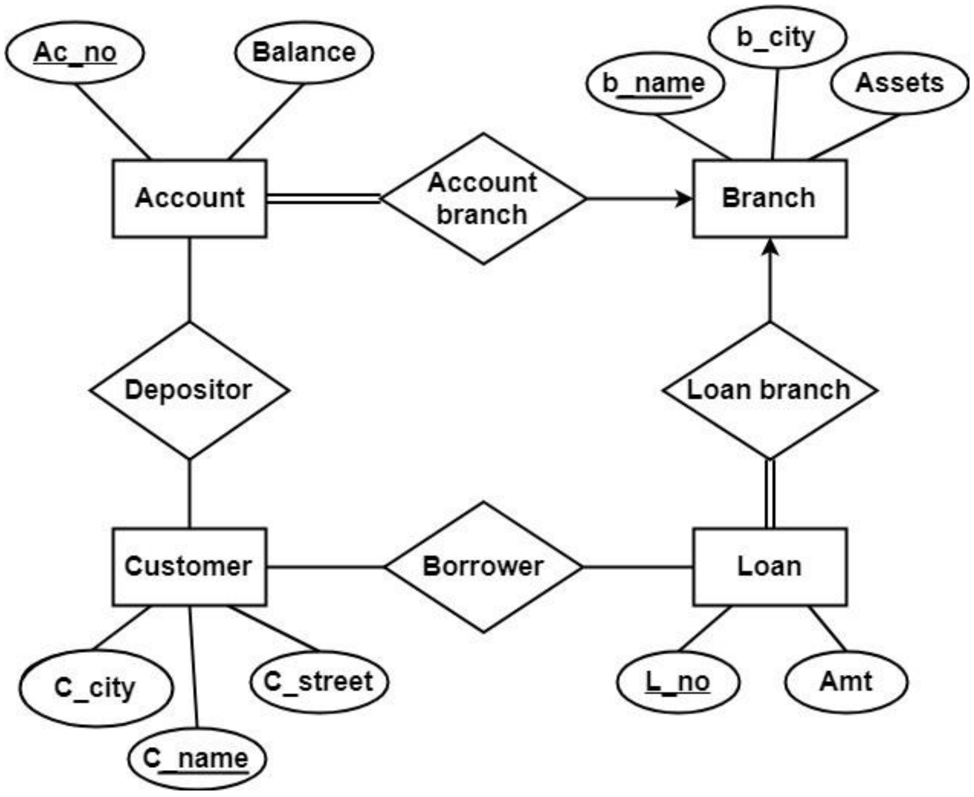




Activity to be Submitted by Students

1. Draw an E-R Diagram For an ATM System.

2. Convert below diagram to tables.

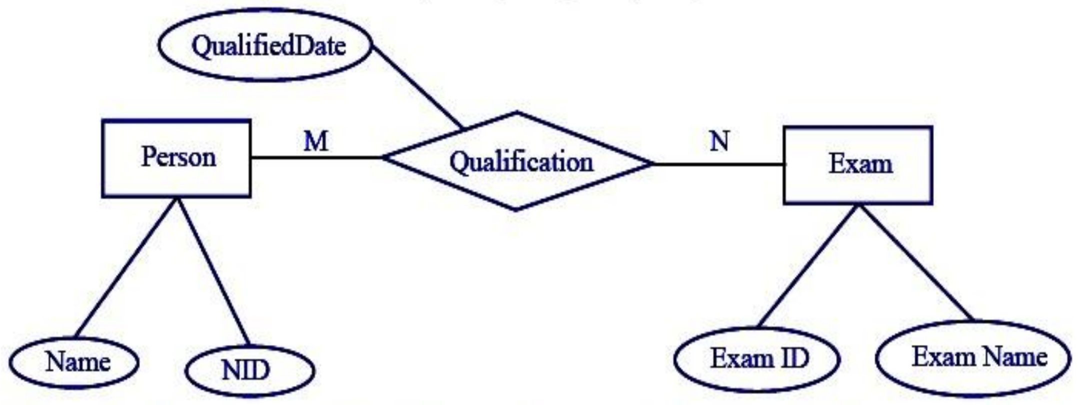


3. Normalize below table till 3 NF

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

4.

Consider the following Relationship Entity Diagram(ERD)



Which of the following possible relations will not hold if the above ERD is mapped into a relation model?

Person (NID, Name)

Qualification (NID, ExamID, QualifiedDate)

Exam (ExamID, NID, ExamName)

Exam (ExamID, ExamName)

CONCLUSION:

Students are able to design ER diagram and convert it into table with Normalized tables.

FAQ :

1. What is an ER Diagram?

An ER diagram is a graphical representation of an entity relationship model. It is used to represent the relationships between entities in a database.

2. Can you explain the different symbols used in an ER diagram?

The three most common symbols used in an ER diagram are rectangles, diamonds, and ovals. Rectangles are used to represent entities, while diamonds represent relationships between those entities. Ovals are used to represent attributes.

3. How do you represent weak entities in ER diagrams?

Weak entities are represented by double rectangles in ER diagrams.

4. What are some of the best practices for designing an ERD?

Some of the best practices for designing an ERD include using a consistent notation, keeping the diagram simple and easy to understand, and using primary and foreign keys to connect related entities.

5. Why are composite keys important when modeling databases with entity-relationship diagrams?

Composite keys are important when modeling databases with entity-relationship diagrams because they can help to uniquely identify each row in a table. This is especially important when modeling databases that will be used for transactional purposes, such as online stores. By using a composite key, you can ensure that each row in the database is uniquely identified, which can help to prevent errors during transactions.

6. What is the difference between a crow's foot and Chen notation? Which one would you recommend based on your experience using them?

The main difference between crow's foot and Chen notation is that crow's foot notation uses boxes to represent entities, while Chen notation uses circles. I would recommend using crow's foot notation because it is generally easier to understand.

7. In what situations should we use ER diagrams vs UML class diagrams?

ER diagrams are used to model the data stored in a database, while UML class diagrams are used to model the structure of a system. If you are trying to model the relationships between the data in a database, then you would use an ER diagram. If you are trying to model the relationships between the different components of a system, then you would use a UML class diagram.

8. How can cardinality be applied to relationships in ER diagrams?

Cardinality is a way of representing the minimum and maximum number of occurrences of an entity in a relationship. For example, if you have a relationship between two entities, A and B, and you want to indicate that A can be related to B multiple times, you would use a cardinality of 1:N.

9. Is it possible to define inheritance hierarchies with ER diagrams?

Yes, it is possible to define inheritance hierarchies using ER diagrams. This is typically done by using a supertype/subtype relationship, where the supertype is the parent class in the inheritance hierarchy and the subtype is the child class.

10. What is normalization?

Normalization is the process of organizing data in a database so that it meets certain requirements, in order to reduce data redundancy and improve data integrity.

11. What is the purpose of creating multiple subtypes or supertypes in an ER diagram?

The purpose of creating multiple subtypes or supertypes in an ER diagram is to provide a more granular level of detail when modeling a specific entity type. This can be useful when trying to track different types of information for a given entity, or when trying to create a more general model that can be applied to multiple entity types.

12. What does it mean to generalize/specialize an object in an ER diagram?

Generalization is the process of creating a more general object from a more specific object. In an ER diagram, this is represented by an arrow going from the more specific object to the more general object. Specialization is the process of creating a more specific object from a more general object. In an ER diagram, this is represented by an arrow going from the more general object to the more specific object.

13. When should I use optionality and why?

Optionality is used to indicate whether an attribute is required or optional. If an attribute is required, then it must be included in every instance of the entity. If an attribute is optional, then it does not have to be included. Optionality is important because it can help to ensure that data is consistent and complete.

14. What is an identifying relationship?

An identifying relationship is a relationship in which the primary key of the parent table is also a foreign key in the child table. This means that the child table cannot exist without the parent table, and the child table cannot be linked to more than one parent table.

15. What do you understand about aggregation, composition, and association?

Aggregation is a relationship between two classes where one class is a part of the other. Composition is a relationship between two classes where one class contains the other. Association is a relationship between two classes where one class is related to the other.

16. What is an attribute? How is it represented in an ER diagram?

An attribute is a characteristic of an entity. In an ER diagram, attributes are represented by oval shapes with the name of the attribute written inside.

17. What is a primary key? How is it represented in an ER diagram?

A primary key is a unique identifier for a given entity. In an ER diagram, it is represented by a bolded, underlined attribute.

18. What is a foreign key? How is it represented in an ER diagram?

A foreign key is a column in a database table that is used to store a reference to the primary key of another table. In an ER diagram, a foreign key is represented by a dashed line that connects the two tables.

19. What is the purpose of data tables that contain only one-to-one relationships?

Data tables that contain only one-to-one relationships are typically used to store data that is not essential to the functioning of the database, but which may be useful for reference purposes. For example, a database for a library might have a table containing information on the authors of the books in the library’s collection. This data would not be essential to the functioning of the database, but it could be useful for reference purposes.

ASSIGNMENT NO: 02

AIM:

SQL Queries:

a. Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.

b. Write at least 10 SQL queries on the suitable database application using SQL DML statements.

Note: Instructor will design the queries which demonstrate the use of concepts like Insert, Select, Update, Delete with operators, functions, and set operator etc.

OBJECTIVES:

Implement SQL queries for given requirements, using different SQL concepts

OUTCOMES :

- Implement SQL queries for given requirements, using different SQL concepts

ENVIRONMENT:

Mysql

THEORY:

Database: - a structured set of data held in a computer, especially one that is accessible in various ways. **Database System**: - A database management system (DBMS) is a computer software application that interactswith the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS isdesigned to allowthe definition, creation, querying, update, and administration of databases.

A database management system (DBMS) is a software tool that makes it possible to organize data in a database.**Adatabase** is a collection of information that is organized so that it can be easily accessed, managed and updated.

SQL: The Language of Database Access: -

Structured Query Language (SQL) is a standardized programming language for accessing and manipulating databases. In an RDBMS like MySQL, Sybase, Oracle, or IMB DM2, SQL writes programming that can manage data and stream data processing. SQL is like a database's own version of a server-side script and is responsible for:

- Executing queries, which are "questions" asked of the database
- Retrieving data
- Editing data: inserting, updating, deleting, or creating new records
- Creating views

- Setting permissions
- Creating new databases

SQL is a standard programming language, but has a number of variations—including some databases' own proprietary SQL extensions.

Top 7 open source relational databases: -

CUBRID, Firebird, MariaDB, MySQL, Postgre SQL, SQLite

RDBMS Terminology

Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.

- **Database** − A database is a collection of tables, with related data.
- **Table** − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column** − One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row** − A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- **Redundancy** − Storing data twice, redundantly to make the system faster.
- **Primary Key** − A primary key is unique. A key value can not occur twice in one table. With a key, youcan only find one row.
- **Foreign Key** − A foreign key is the linking pin between two tables.
- **Compound Key** − A compound key (composite key) is a key that consists of multiple columns, becauseone column is not sufficiently unique.
- **Index** − An index in a database resembles an index at the back of a book.
- **Referential Integrity** − Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

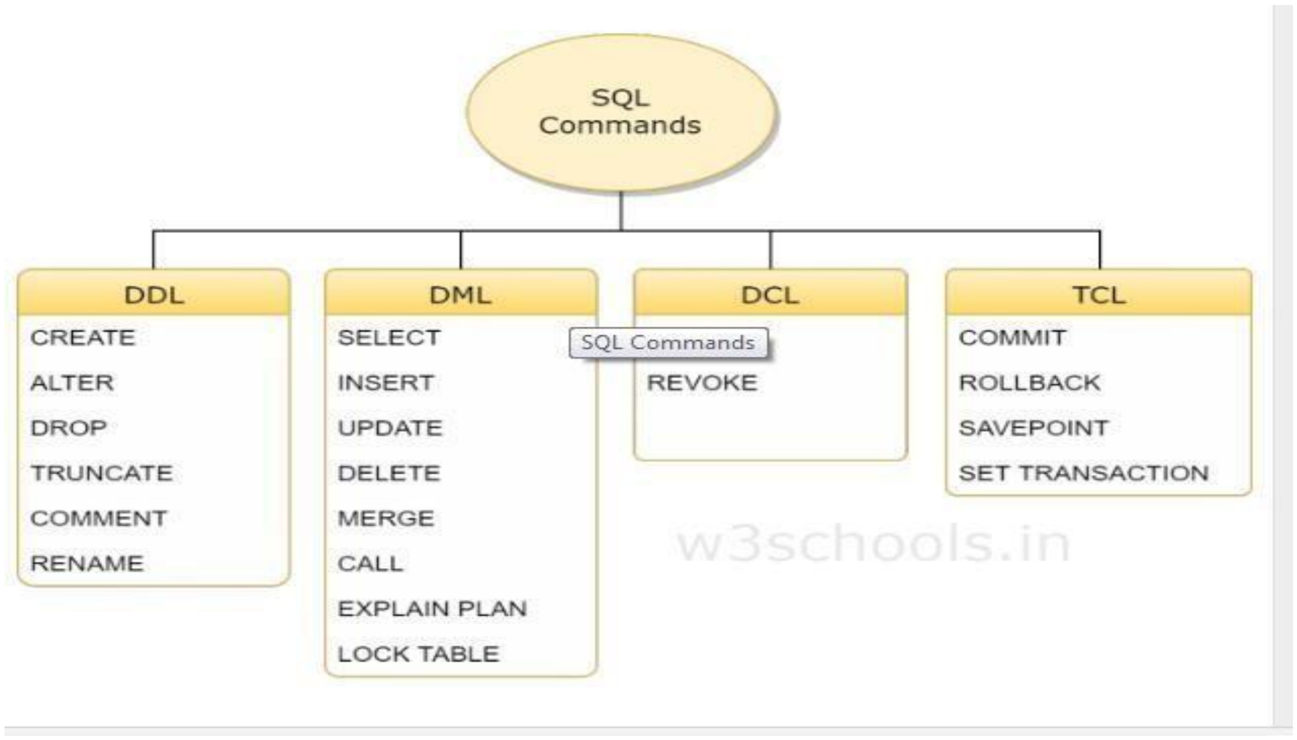
- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.

- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

Administrative MySQL Command

Here is the list of the important MySQL commands, which you will use time to time to work with MySQL database −

- **USE Databasename** − This will be used to select a database in the MySQL workarea.
- **SHOW DATABASES** − Lists out the databases that are accessible by the MySQL DBMS.
- **SHOW TABLES** − Shows the tables in the database once a database has been selected with the use command.
- **SHOW COLUMNS FROM *tablename:*** Shows the attributes, types of attributes, key information, whether NULL is permitted, defaults, and other information for a table.
- **SHOW INDEX FROM tablename** − Presents the details of all indexes on the table, including the PRIMARY KEY.



MySQL Connection Using MySQL Binary

You can establish the MySQL database using the **mysql** binary at the command prompt.

Example

Here is a simple example to connect to the MySQL server from the command prompt −[root@host]#mysql-u root -p

Enter password:******

This will give you the mysql> command prompt where you will be able to execute any SQL command. Following is the result of above command −

The following code block shows the result of above code –

Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 2854760 to server version: 5.0.9

Type 'help;' or '\h' for help.Type '\c' to clear the buffer.

In the above example, we have used **root** as a user but you can use any other user as well. Any user will be able to perform all the SQL operations, which are allowed to that user.

You can disconnect from the MySQL database any time using the **exit** command at mysql> prompt.mysql> exit

Bye

USE DatabaseName;

Always the database name should be unique within the RDBMS.

DROP DATABASE databaseName -- Delete the database (irrecoverable!) DROP DATABASE IF EXISTS databaseName -- Delete if it exists

CREATE DATABASE databaseName -- Create a new database

CREATE DATABASE IF NOT EXISTS databaseName -- Create only if it does not exists SHOW DATABASES -- Show all the databases in this server

USE databaseName -- Set the default (current) databaseSELECT DATABASE() -- Show the default database

SHOW CREATE DATABASE databaseName -- Show the CREATE DATABASE statement

-- Table-Level

DROP TABLE [IF EXISTS] tableName, ...

CREATE TABLE [IF NOT EXISTS] tableName (

columnName columnType columnAttribute, ...

PRIMARY KEY(columnName),

FOREIGN KEY (columnNmae)
REFERENCES tableName (columnName)
)

SHOW TABLES -- Show all the tablesin the default databaseDESCRIBE|DESC tableName -- Describe the details for a table

ALTER TABLE tableName ... -- Modify a table, e.g., ADD COLUMN and DROP COLUMNALTER TABLE tableName ADD columnDefinition ALTER TABLE tableName DROP columnName

ALTER TABLE tableName ADD FOREIGN KEY (columnNmae) REFERENCES tableName (columnNmae)ALTER TABLE tableName DROP FOREIGN KEY constraintName

SHOW CREATE TABLE tableName -- Show the CREATE TABLE statement for thistableName

Data Manipulation Language (DML)

Allows you to modify the database instance by inserting, modifying, and deleting its data. It is responsible for performing all types of data modification in a database. There are three basic constructs which allow database program and user to enter data and information are:

Here are some important DML commands in SQL:

INSERT UPDATEDELETE

INSERT:

This is a statement is a SQL query. This command is used to insert data into the row of a table.Syntax:

INSERT INTO TABLE_NAME (col1, col2, col3, col N)

VALUES (value1, value2, value3,.valueN);

Or

INSERT INTO TABLE_NAME

VALUES (value1, value2, value3,.valueN);

For example:

INSERT INTO students (RollNo, FIrstName, LastName) VALUES ('60', 'Tom', Erichsen');

UPDATE:

This command is used to update or modify the value of a column in the table.Syntax:

UPDATE table_name SET [column_name1= value1,. column_nameN = valueN] [WHERE

CONDITION]

For example:

UPDATE students

SET FirstName = 'Jhon', LastName= 'Wick'WHERE StudID = 3;

DELETE:

This command is used to remove one or more rows from a table.Syntax:

DELETE FROM table_name [WHERE condition];For example:

DELETE FROM studentsWHERE FirstName = 'Jhon';

I. Set Operators

Set operators combine the results of two component queries into a single result. Queriescontaining set operators are called compound queries. The lists of SQL set operators are "UNION [ALL], INTERSECT, MINUS Operators".You can combine multiple queries usingthe set operators UNION, UNION ALL, INTERSECT, and MINUS. All set operators have equal precedence. If a SQL statement contains multiple set operators, then Oracle Databaseevaluates them from the left to right unless parentheses explicitly specify another order.Thecorresponding expressions in the select lists of the component queriesof a compound querymust match in number and must be in the same datatype group (such as numeric or character).

If component queries select character data, then the datatype of the return valuesaredetermined as follows:

- If both queries select values of datatype CHAR of equal length, then the returned values have datatype CHAR of that length. If the queries select values of CHAR with different lengths, then the returned value is VARCHAR2 with the length of the larger CHAR value.
- If either or both of the queries select values of datatype VARCHAR2, then the returned values have datatype VARCHAR2.

If component queries select numeric data, then the datatype of the return values is determinedby numeric precedence:

- If any query selects values of type BINARY_DOUBLE, then the returned values have datatype BINARY_DOUBLE.
- If no query selects values of type BINARY_DOUBLE but any query selects values of type BINARY_FLOAT, then the returned values have datatype BINARY_FLOAT.
- If all queries select values of type NUMBER, then the returned values havedatatype NUMBER.

In queries using set operators, Oracle does not perform implicit conversion across datatype groups. Therefore, if the corresponding expressions of component queries resolve to both character data and numeric data, Oracle returns an error to fetch onlyunique records insteadof fetching duplicate records.

II. SQL Functions

SQL functions are built into Oracle Database and are available for use in various appropriate SQL statements. Do not confuse SQL functions with user-defined functions written in PL/SQL.If you call a SQL function with an argument of a datatype other than the datatype expected by the SQL function, then Oracle attempts to convert the argument to the expected datatype before performing the SQL function. If you call a SQL function with a null argument, then the SQL function automatically returns null.

1. Single-Row Functions

Single-row functions return a single result row for every row of a queried table or view.

These functions can appear in select lists, WHERE clauses, STARTWITH and CONNECT BY clauses, and HAVING clauses.

2. Character Functions Returning Character Value

Character functions that return character values return values of the following datatypesunless otherwise documented:

- If the input argument is CHAR or VARCHAR2, then the value returnedis VARCHAR2.
- If the input argument is NCHAR or NVARCHAR2, then the value returnedis NVARCHAR2.

The length of the value returned by the function is limited by the maximum length of thedatatype returned.

- For functions that return CHAR or VARCHAR2, if the length of the return value exceeds the limit, then Oracle Database truncates it and returns the result without an error message.
- For functions that return CLOB values, if the length of the return values exceeds the limit, then Oracle raises an error and returns no data.

3. Datetime Functions

Datetime functions operate on date (DATE),timestamp(TIMESTAMP, TIMESTAMP WITH TIME ZONE, and TIMESTAMP WITH LOCAL ZONE), and interval (INTERVAL DAY TO SECOND, INTERVAL YEAR values.

Some of the datetime functions were designed for the Oracle DATE datatype (ADD_MONTHS, CURRENT_DATE, LAST_DAY, NEW_TIME, and NEXT_DAY).

If you provide a timestamp value as their argument, Oracle Database internally converts the input type toa DATE value and returns a DATE value. The exceptions are the MONTHS_BETWEEN function,which returns a number, and the ROUND and TRUNC functions, which do not accept timestamp or interval values at all. The remaining datetime functions were designed to accept any of the three types of data (date, timestamp, and interval) and to return a value of one of these types.

Activity to be Submitted by Students

(Insert, Select, Update, Delete, operators, functions, setoperator, all constraints, view, index, synonym, sequence)

Create following relations and excecute the queries given below.
Account(Acc_no, branch_name,balance)
branch(branch_name,branch_city,assets)
customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)

Solve following queries:
Q1. Find the names of all branches in loan relation.
Q2. Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000.
Q3. Find all customers who have a loan from bank. Find their names,loan_no and loan amount.
Q4. List all customers in alphabetical order who have loan from Akurdi branch.
Q5. Find all customers who have an account or loan or both at bank.
Q6. Find all customers who have both account and loan at bank.
Q7. Find all customer who have account but no loan at the bank.
Q8. Find average account balance at Akurdi branch.
Q9. Find the average account balance at each branch
Q10. Find no. of depositors at each branch.
Q11. Find the branches where average account balance > 12000.
Q12. Find number of tuples in customer relation.
Q13. Calculate total loan amount given by bank.
Q14. Delete all loans with loan amount between 1300 and 1500.
Q15. Delete all tuples at every branch located in Nigdi.
Q.16. Create synonym for customer table as cust.
Q.17. Create sequence roll_seq and use in student table for roll_no column.
Create above tables with appropriate constraints like primary key, foreign key, check constrains, not null etc.

CONCLUSION:

Students are able to implement SQL queries for given requirements, using different SQL concepts.

FAQ:

1. **What do DDL, DML, and DCL stand for?**

DDL is the abbreviation for Data Definition Language dealing with database schemas, as well as the description of how data resides in the database. An example of this is the CREATE TABLE command. DML denotes Data Manipulation Language which includes commands such as SELECT, INSERT, etc. DCL stands for Data Control Language and includes commands like GRANT, REVOKE, etc.

2. **What are the common MySQL functions?**

Common MySQL functions are as follows:

- **ABS():** Returns the absolute value of a number. It removes the negative sign if the number is negative.

- **ROUND():** Rounds a number to a specified number of decimal places. It can round to the nearest integer or a specific decimal position.
- **CEIL():** Returns the smallest integer greater than or equal to a given number. It rounds up the value to the nearest integer.
- **FLOOR():** Returns the largest integer less than or equal to a given number. It rounds down the value to the nearest integer.
- **EXP():** Calculates the exponential value of a number. It returns the result of raising the mathematical constant e to the power of the given number.
- **LOG():** Calculates the natural logarithm of a number. It returns the logarithm base e (natural logarithm) of the given number.
- **NOWO:** The function for returning the current date and time as a single value
- **CURRDATEO:** The function for returning the current date or time
- **CONCAT (X, Y):** The function to concatenate two string values creating a single string output
- **DATEDIFF (X, Y):** The function to determine the difference between two dates

3. What is the difference between CHAR and VARCHAR?

When a table is created, CHAR is used to define the fixed length of the table and columns. The length value could be in the range of 1–255. The VARCHAR command is used to adjust the column and table lengths as required.

4. What is the syntax for concatenating tables in MySQL?

The syntax for concatenating tables in MySQL:

CONCAT (string 1, string 2, string 3)

5. What is the limit of indexed columns that can be created for a table?

It depends on the storage engine used:

For the MyISAM storage engine, the limit is 64 while for the InnoDB storage engine, the limit is 16.

6. What are the different types of strings used in database columns in MySQL?

In MySQL, the different types of strings that can be used for database columns are SET, BLOB, VARCHAR, TEXT, ENUM, and CHAR.

7. How can a user get the current SQL version?

The syntax for getting the current version of MySQL:

SELECT VERSION ();

8. What is the difference between primary key and unique key?

While both are used to enforce the uniqueness of the column defined, the primary key would create a clustered index, whereas the unique key would create a non-clustered index on the column. The primary key does not allow 'NULL', but the unique key does.

9. What is the difference between the primary key and the candidate key?

The primary key in MySQL is used to identify every row of a table in a unique manner. For one table, there is only one primary key. The candidate keys can be used to reference the foreign keys. One of the candidate keys is the primary key.

10. What are the differences between a primary key and a foreign key?

Primary Key	**Foreign Key**
It helps in the unique identification of data in a database	It helps establish a link between tables
There can be only one primary key for a table	There can be more than one foreign key for a table
Primary key attributes cannot have duplicate values in a table	Duplicate values are acceptable for a foreign key
Null values are not acceptable	Null values are acceptable
We can define primary key **constraints** for temporarily created tables	It cannot be defined for temporary tables
The primary key index is automatically created	The index is not created automatically

11. What is the use of ENUM in MySQL?

The use of ENUM will limit the values that can go into a table. For instance, a user can create a table giving specific month values and other month values would not enter into the table.

12. What is the difference between the DELETE TABLE and TRUNCATE TABLE commands in MySQL?

Basically, DELETE TABLE is a logged operation, and every row deleted is logged. Therefore, the process is usually slow. TRUNCATE TABLE also deletes rows in a table, but it will not log any of the rows deleted. The process is faster here in comparison. TRUNCATE TABLE can be rolled back and is functionally similar to the DELETE statement without a WHERE clause.

13. What is the TIMESTAMP data type?

Timestamp in SQL Server helps in row versioning. Row versioning is a type of concurrency that allows retaining the value until it is committed in the database. It shows the instant time of any event. It consists of both the date and time of the event. Also, timestamp helps in backing up data during the failure of a transaction.

While we insert, update, or delete a record, the date and time automatically get inserted.

- Format of timestamp: YYYY-MM-DD HH:MM: SS
- Range of timestamp: “1970-01-01 00:00:01” UTC to “2038-01-19 03:14:07” UTC

ASSIGNMENT NO: 03

AIM:

SQL Queries – all types of Join, Sub-Query and View:

Write at least10 SQL queries for suitable database application using SQL DML statements. Note: Instructor will design the queries which demonstrate the use of concepts like all types ofJoin, Sub-Query and View

OBJECTIVES:

Implement SQL queries for given requirements, using different SQL concepts

OUTCOMES :

- Implement SQL queries for given requirements, using different SQL concepts

ENVIRONMENT:

Mysql

THEORY:

Join in SQL

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set ofdata. SQL Join is used for combining column from two or more tables by using values common to both tables. **Join** Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is **(n-1)** where **n**, is number of tables. A table can also join toitself known as, **Self Join**.
Types of Join:-

The following are the types of JOIN that we can use in SQL.

- Inner
- Outer
- Left
- Right

Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return atable which consists of records which combines each row from the first table with each row ofthe second table.
Cross JOIN Syntax is, SELECT column-name-listfrom *table-name1*

CROSS JOIN

table-name2;

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition

specified in the query. Inner Join Syntax is, SELECT column-name-listfrom *table-name1*

INNER JOIN

table-name2

WHERE table-name1.column-name = table-name2.column-name;

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and samedatatype present in both the tables to be joined.
Natural Join Syntax is,
SELECT *

from *table-name1*

NATURAL JOIN

table-name2;

Outer JOIN

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

- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Outer Join

The left outer join returns a result table with the **matched data** of two tables then remainingrows of the **left** table and null for the **right** table's column.
Left Outer Join syntax is, SELECT column-name-listfrom *table-name1*

LEFT OUTER JOIN

table-name2

on table-name1.column-name = table-name2.column-name;Left outer Join Syntax for **Oracle** is,
select column-name- listfrom *table-name1*,*table-name2*

on table-name1.column-name = table-name2.column-name(+);

Left Outer Join query will be,

SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id=class_info.id);The result table will look like,

ID	NAME	ID	ADDRESS
1	Abhi	1	DELHI
2	Adam	2	MUMBAI
3	Alex	3	CHENNAI
4	Anu	Null	Null
5	Ashish	Null	Null

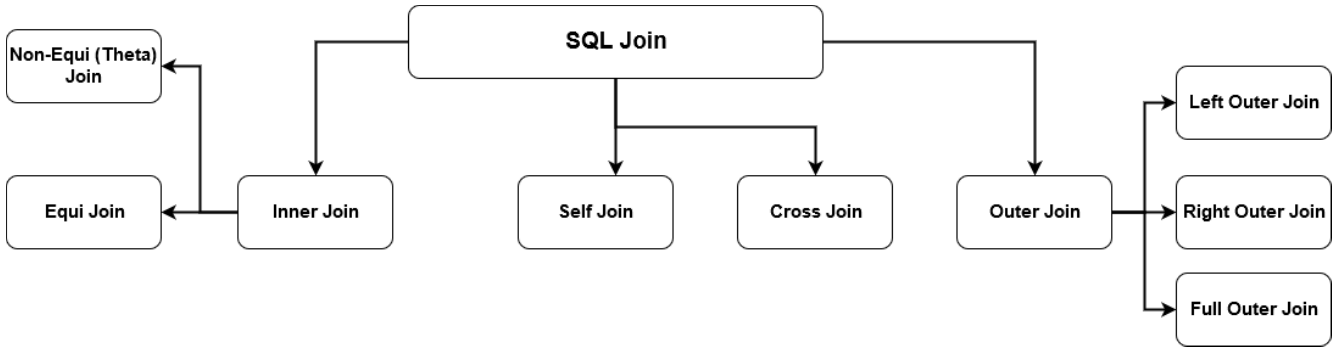
Right Outer Join

The right outer join returns a result table with the **matched data** of two tables then remaining rows of the **right table** and null for the **left** table's columns.
select column-name-listfrom *table-name1* **RIGHT OUTER JOIN**

table-name2

on table-name1.column-name = table-name2.column-name;Right outer Join Syntax for **Oracle** is,
select column-name- listfrom *table-name1*,*table-name2*

on table-name1.column-name(+) = table-name2.column-name;



SQL Subquery

Subquery or **Inner query** or **Nested query** is a query in a query. SQL subquery is usually addedin the WHERE Clause of the SQL statement. Most of the time, a subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value in the database.

Subqueries are an alternate way of returning data from multiple tables.

Subqueries can be used with the following SQL statements along with the comparision operators like =, <, >, >=, <= etc.

- SELECT
- INSERT
- UPDATE
- DELETE

SQL Subquery Example:

Usually, a subquery should return only one record, but sometimes it can also return multiple records when used with operators LIKE IN, NOT IN in the where clause. Thequery syntax would be like,

SELECT first_name, last_name, subjectFROM student_details

WHERE games NOT IN ('Cricket', 'Football');

Subquery output would be similar to:

```
first_nam     last_na     subject
e             me
-------------------------------------------------------
Shekar        Gowda       Badminton

Priya         Chandra     Chess
```

SQL Subquery INSERT Statement

Subquery can be used with INSERT statement to add rows of data from one or more tables toanother table. Lets try to group all the students who study Maths in a table 'maths_group'. INSERT INTO maths_group(id, name)

SELECT id, first_name || ' ' || last_name
FROM student_details WHERE subject= 'Maths'

SQL Subquery SELECT Statement

A subquery can be used in the SELECT statement as follows. Lets use the product and order_items table defined in the sql_joins section.

select p.product_name, p.supplier_name, (select order_id from order_items where product_id = 101) as order_id from product p where p.product_id = 101

```
product_na       supplier_       orde
me               name            r_id
------------------------------------------------
Television       Onida           5103
```

Correlated Subquery

A query is called correlated subquery when both the inner query and the outer query are interdependent. For every row processed by the inner query, the outer query is processed as well.The inner query depends on the outer query before it can be processed.
SELECT p.product_name FROM product p

WHERE p.product_id = (SELECT o.product_id FROM order_items oWHERE o.product_id = p.product_id);

Subquery Notes **Nested Subquery**

1) You can nest as many queries you want but it is recommended not to nest more than 16subqueries in oracle

Non-Corelated Subquery

2) If a subquery is not dependent on the outer query it is called a non-correlated subquery

Subquery Errors

3) Minimize subquery errors: Use drag and drop, copy and paste to avoid running subqueries with spelling and database typos. Watch your multiple field SELECT comma use, extra or to few getting SQL error message "Incorrect syntax".

SQL Subquery Comments

Adding SQL Subquery comments are good habit (/* your command comment */) which can saveyou time,clarify your previous work .. results in less SQL headaches.

SQL Views

A VIEW is a virtual table, through which a selective portion of the data from one or more tablescan be seen.Views do not contain data of their own. They are used to restrict access to the database or to hide data complexity. A view is stored as a SELECT statement in the database.
DML operations on a view like INSERT, UPDATE, DELETE affects the data in the originaltable uponwhich the view is based.

The Syntax to create a sql view is

CREATE VIEW view_nameASSELECT column_list
FROM table_name [WHERE condition];

- ***view_name*** is the name of the VIEW.
- The SELECT statement is used to define the columns and rows that you want to displayin the view.
- **For Example:** to create a view on the product table the sql query would be like,CREATEVIEW view_product AS SELECT product_id, product_nameFROMproduct;

Students are able to implement SQL queries all types of Join, Sub-Query and View for given requirements, usingdifferent SQL concepts

Activity to be Submitted by Students

Design SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

1. Create following Tables

cust_mstr(cust_no,fname,lname)

add_dets(code_no,add1,add2,state,city,pincode)

Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'

2.Create following Tables

cust_mstr(custno,fname,lname)

acc_fd_cust_dets(codeno,acc_fd_no)

fd_dets(fd_sr_no,amt)

List the customer holding fixed deposit of amount more than 5000

3. Create following Tables

emp_mstr(e_mpno,f_name,l_name,m_name,dept,desg,branch_no)

branch_mstr(name,b_no)

List the employee details along with branch names to which they belong

4. Create following Tables

emp_mstr(emp_no,f_name,l_name,m_name,dept)

cntc_dets(code_no,cntc_type,cntc_data)

List the employee details along with contact details using left outer join & right join

5. Create following Tables

cust_mstr(cust_no,fname,lname)

add_dets(code_no,pincode)

List the customer who do not have bank branches in their vicinity.

6.

a) Create View on borrower table by selecting any two columns and perform insert update

delete operations

b) Create view on borrower and depositor table by selecting any one column from each table

perform insert update delete operations

c) create updateable view on borrower table by selecting any two columns and perform insert, update and delete operations.

CONCLUSION :

Students are able to implement SQL queries all types of Join, Sub-Query and View for given requirements, usingdifferent SQL concepts.

FAQ:

1. **What are joins in SQL?**

A join clause is a SQL command used to combine records from multiple tables or retrieve data from these tables based on the existence of a common field (column) between them. A join condition and SELECT statement can be used to join the tables. Using the SQL JOIN clause, records can be fetched from two or more tables in a database and combined. In general, they are used when users need to retrieve data from tables that contain many-to-many or one-to-many relationships between them.

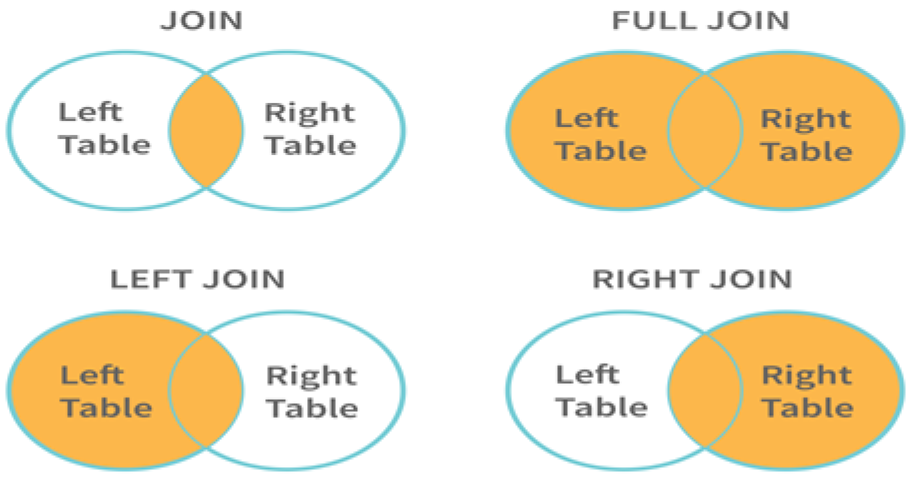
2. What is the importance of SQL joins in database management?

SQL joins are important in database management for the following reasons:

- A method of stitching a database back together to make it easier to read and use.
- Additionally, they maintain a normalized database. Data normalization helps us keep data redundancy low so that when we delete or update a record, we will have fewer data anomalies in our application.
- Joins have the advantage of being faster, and as a result, are more efficient.
- It is almost always faster to retrieve the data using a join query rather than one that uses a subquery.
- By utilizing joins, it is possible to reduce the workload on the database. For example, instead of multiple queries, you can use one join query. So, you can better utilize the database's ability to search, filter, sort, etc.

3. What are the different types of Joins in SQL?

There are various types of join statements you can use depending on the use case you have. Specifically, there are four types of joins as follows:



- **Inner Join:** This method returns datasets that have the same values in both tables.
- **Full Join:** This is also referred to as a full outer join. It combines all rows from the left table and the right table to create the result set. It means that a query would return records from both tables, even if they had NULL values. If there are no matching rows, the result-set (joined table) will display NULL values.
- **Right Join:** This is also referred to as the Right outer join. All records from the right table will be returned, as well as any records that match from the left table.
- **Left Join:** This is also referred to as Left outer join. All records from the left table will be returned, as well as any records that match from the right table.

4. Explain merge join in SQL.

Merge join produces a single output stream resulting from the joining of two sorted datasets using an INNER, FULL, or LEFT join. It is the most effective of all the operators for joining data. Specifically, merge join requires that both inputs be sorted as well as matching meta-data in the joined columns. Users can't join columns of different data types together. Users are not permitted to combine a column with a numeric data type with a column with a character data type.

5. State the difference between inner join and left join.

- **Inner Join:** This join generates datasets that contain matching records in both tables (left and right). By using an inner join, only the rows that match between each of the tables are returned; all non-matching rows are removed.

Left Join: It returns datasets that have matching records in both tables (left and right) plus non-matching rows from the left table. By using a left join, all the records in the left table plus the matching records in the right table are returned.

6. State difference between left join and right join.

- **Left Join:** It returns datasets that have matching records in both tables (left and right) plus non-matching rows from the left table. By using a left join, all the records in the left table plus the matching records in the right table are returned.
- **Right Join:** It returns datasets that have matching records in both tables (left and right) plus non-matching rows from the right table. By using a left join, all the records in the right table plus the matching records in the left table are returned.

ASSIGNMENT NO: 04

AIM:

Implement PL/SQL Code block for given requirements.

Consider Tables:

1. Borrower(Roll_no, Name, Date of Issue, Name of Book, Status)

2. Fine(Roll_no, Date, Amt)

• Accept Roll_no and Name of Book from user.

• Check the number of days (from date of issue).

• If days are between 15 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.

OBJECTIVES:

Implement PL/SQL Code block for given requirements.

OUTCOMES :

Implement PL/SQL Code block for given requirements.

ENVIRONMENT:

Mysql

THEORY:

A. Control Structures: - PL/SQL allows the use of an IF statement to control the execution of a block of code. In PL/SQL, the IF -THEN - ELSIF - ELSE - END IF construct in code blocks allow specifying certain conditions under which a specific block of code should be execute PL/SQL Control Structures are used to control flow of execution. PL/SQL provides different kinds of statements to provide such type of procedural capabilities. These statements are almost same as that of provided by other languages. The flow of control statements can be classified into the following categories:

- Conditional Control
- Iterative Control
- Sequential Control

Conditional Control:

PL/SQL allows the use of an IF statement to control the execution of a block of code. In PL/SQL, the IF -THEN - ELSIF - ELSE - END IF construct in code blocks allow specifying certain conditions under which a specific block of code should be executed.

```
IF < Condition > THEN

< Action >

ELSIF <Condition> THEN

   < Action > ELSE < Action >END IF;
```

Iterative Control :

Iterative control indicates the ability to repeat or skip sections of a code block. A **loop** marks a sequence of statements that has to be repeated. The keyword loop has to be placed before the first statement in the sequence of statements to be repeated, while the keyword end loop is placed immediately after the last statement in the sequence. Once a loop begins to execute, it will go on forever. Hence a conditional statement that controls the number of times a loop is executed always accompanies loops. PL/SQL supports the following structures for iterative control:

Simple loop : In simple loop, the key word loop should be placed before the first statement in the sequence and the keyword end loop should be written at the end of the sequence to end the loop.

Syntax:

```
Loop


 < Sequence of statements >End loop;
```

1. WHILE loop

The while loop executes commands in its body as long as the condition remains true

Syntax:

```
WHILE < condition >LOOP

< Action >
```

2. The FOR Loop

The FOR loop can be used when the number of iterations to be executed are known.

Syntax :

```
FOR variable IN [REVERSE] start..endLOOP
< Action >END LOOP;
```

The variable in the For Loop need not be declared. Also the increment value cannot be specified.The For Loop variable is always incremented by 1.

3. Sequential Control :

The GOTO Statement

The GOTO statement changes the flow of control within a PL/SQL block. This statement allows execution of a section of code, which is not in the normal flow of control. The entry point into such a block of code is marked using the tags «userdefined name». The GOTO statement can then make use of this user-defined name to jump into that block of code for execution.

Syntax :

```
GOTO jump;
....
<<jump>>
```

B. Exceptions

An Exception is an error situation, which arises during program execution. When an error occurs exception is raised, normal execution is stopped and control transfers to exception handling part. Exception handlers are routines written to handle the exception. The exceptions can be internally defined (system-defined or pre-defined) or User-defined exception.

Syntax:

```
EXCEPTION
WHEN <ExceptionName> THEN
<User Defined Action To Be Carried Out>
```

Predefined exception:

Predefined exception is raised automatically whenever there is a violation of Oracle coding rules. Predefined exceptions are those like ZERO_DIVIDE, which is raised automatically when we try to divide a number by zero. Other built-in exceptions are given below. You can handle unexpected Oracle errors using OTHERS handler. It can handle all raised exceptions that are not handled by any other handler. It must always be written as the last handler in exception block.

Exception	Raised when....
DUP_VAL_ON_INDEX	When you try to insert a duplicate value into a unique column.
INVALID_CURSOR	It occurs when we try accessing an invalid cursor.
INVALID_NUMBER	On usage of something other than number in place of number value.
LOGIN_DENIED	At the time when user login is denied.
TOO_MANY_ROWS	When a select query returns more than one row and the destination variable can take only single value.
VALUE_ERROR	When an arithmetic, value conversion, truncation, or constraint error occurs.
CURSOR_ALREADY_OPEN	Raised when we try to open an already open cursor.

Predefined exception handlers are declared globally in package STANDARD. Hence we need not have to define them rather just use them. The biggest advantage of exception handling is it improves readability and reliability of the code. Errors from many statements of code can be handles with a single handler. Instead of checking for an error at every point we can just add an exception handler and if any exception is raised it is handled by that. For checking errors at a specific spot it is always better to have those statements in a separate begin – end block.

User Defined Exception Handling:

To trap business rules being violated the technique of raising user-defined exceptions and then handling them, is used. User-defined error conditions must be declared in the declarative part of any PL/SQL block. In the executable part, a check for the condition that needs special attention is made. If that condition exists, the call to the user-defined exception is made using a RAISE statement. The exception once raised is then handled in the Exception handling section of the PL/SQL code block.

Syntax

```
DECLARE
   < ExceptionName > EXCEPTION ;BEGIN
<SQL Sentence >;
```

Activity to be Submitted by Students

Problem Statement
Borrow(Roll_no, Name, DateofIssue, NameofBook, Status)
Fine(Roll_no,Date,Amt)
Accept roll_no & name of book from user. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table. Also handles the exception by named exception handler or user define exception

CONCLUSION:

Students are able to Implement PL/SQL Code block for given requirements

FAQ :

1. **What are the features of PL/SQL?**

Following are the features of PL/SQL:

- PL/SQL provides the feature of decision making, looping, and branching by making use of its procedural nature.
- Multiple queries can be processed in one block by making use of a single command using PL/SQL.
- The PL/SQL code can be reused by applications as they can be grouped and stored in databases as PL/SQL units like functions, procedures, packages, triggers, and types.
- PL/SQL supports exception handling by making use of an exception handling block.
- Along with exception handling, PL/SQL also supports error checking and validation of data before data manipulation.
- Applications developed using PL/SQL are portable across computer hardware or operating system where there is an Oracle engine.

2. What do you understand by PL/SQL table?

- PL/SQL tables are nothing but objects of type tables that are modeled as database tables. They are a way to provide arrays that are nothing but temporary tables in memory for faster processing.
- These tables are useful for moving bulk data thereby simplifying the process.

3. Explain the basic structure followed in PL/SQL?

- The basic structure of PL/SQL follows the BLOCK structure. Each PL/SQL code comprises SQL and PL/SQL statement that constitutes a PL/SQL block.
- Each PL/SQL block consists of 3 sections:
  - The optional Declaration Section
  - The mandatory Execution Section
  - The optional Exception handling Section

4. Explain control structures used in PL/SQL.

PL/SQL IF Statements

1. IF-THEN Statement

Syntax

IF condition

THEN

Statement;

END IF;

This syntax is used when user needs to execute statements when condition is true.
2. IF-THEN-ELSE Statement

Syntax

IF condition

THEN

[Statements to execute when condition is TRUE]

ELSE

[Statements to execute when condition is FALSE]

END IF;

This syntax is used to execute one set of statements when condition is TRUE or different set of statements when condition is FALSE.

4. IF-THEN-ELS-IF-ELSE Statement

Syntax

```
IF condition1
THEN
Statements to execute when condition1 is TRUE
ELSIF condition2
THEN
Statements to execute when condition2 is TRUE
ELSE
Statements to execute when both condition1 and condition2 are FALSE
END IF;
```

This syntax is used to execute one set of statements if condition1 is TRUE, a different set of statements when condition2 is TRUE or a third set of statements when both condition1 and condition2 are false.

PL/SQL Case statement

The PL/SQL CASE Statement provides facility to execute a sequence of statements based on a selector. A selector may be variable, function or an expression.

Syntax

```
CASE [expression]
WHEN condition1 THEN result1
WHEN condition2 THEN result2
…....................
WHEN condition_n THEN result_n
ELSE result
END;
```

ASSIGNMENT NO .5

AIM: Named PL/SQL Block: 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 and900 category is first class, if marks 899 and 825 category is Higher Second Class.

COURSE OBJECTIVE:
Implement PL/SQL Code block for given requirements

COURSE OUTCOME:

Implement PL/SQL Code block for given requirements

ENVIRONMENT: - Mysql

THEORY:
Stored Procedures : A stored procedure or in simple a proc is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages. A procedure has a header anda body. The header consists of the name of the procedure and the parameters or variables passed to the procedure. The body consists or declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.
Procedures: Passing Parameters
We can pass parameters to procedures in three ways.
1) IN-parameters
2) OUT-parameters
3) IN OUT-parameters
A procedure may or may not return any value.
General Syntax to create a procedure is:

```
CREATE [OR REPLACE] PROCEDURE  proc_name  [list of parameters] IS Declaration section

BEGIN
Execution  section

EXCEPTION
Exception section
END;
```

IS - marks the beginning of the body of the procedure and is similar to DECLARE in anonymous PL/SQL Blocks. The code between IS and BEGIN forms the Declaration section.

The syntax within the brackets [ ] indicate they are optional. By using CREATE OR REPLACE together the

procedure is created if no other procedure with the same name exists or the existing procedure is replaced with the current code.
Procedures: Example
The below example creates a procedure ‘employer_details’ which gives the details of the employee.

```
1>	CREATE OR REPLACE PROCEDURE employer_details
2>	IS
3>	CURSOR emp_cur IS
4>	 SELECT first_name, last_name, salary FROM emp_tbl;5> emp_rec emp_cur%rowtype;
6>	BEGIN
7>	FOR emp_rec in sales_cur8> LOOP
9>	dbms_output.put_line(emp_cur.first_name || ' ' ||emp_cur.last_name10>   || ' ' ||emp_cur.salary); 11>
	END LOOP;
12>	END;
```

There are two ways to execute a procedure.
1) **From the SQL prompt.**
EXECUTE [or EXEC] procedure_name;

2) **Within another procedure – simply use the procedure name.**

procedure_name;

PL/SQL Functions
A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedureand a function is, a function must always return a value, but a procedure may or may not return a value.
General Syntax to create a function is –

```
CREATE [OR REPLACE] FUNCTION function_name [parameters] RETURN return_datatype;
IS
Declaration_section

BEGIN
Execution_section

Return return_variable;


EXCEPTION
exception section Return return_variable; END;
```

1) Return Type: The header section defines the return type of the function. The return datatype can be anyof the oracle datatype like varchar, number etc.

2) The execution and exception section both should return a value which is of the datatype defined in theheader section.

For example, let’s create a frunction called "employer_details_func' similar to the one created in stored proc1>

1>CREATE OR REPLACE FUNCTION employer_details_func
2> RETURN VARCHAR(20);
3> IS
5> emp_name VARCHAR(20);
6> BEGIN
7> SELECT first_name INTO emp_name 8> FROM emp_tbl WHERE empID = '100';9> RETURN emp_name;
10> END;

In the example we are retrieving the ‘first_name’ of employee with empID 100 to variable ‘emp_name’. The return type of the function is VARCHAR which is declared in line no 2.

The function returns the 'emp_name' which is of type VARCHAR as the return value in line no
9.A function can be executed in the following ways.

1) Since a function returns a value we can assign it to a variable.employee_name := employer_details_func;

If ‘employee_name’ is of datatype varchar we can store the name of the employee by assigning the return type ofthe function to it.

2) As a part of a SELECT statement

SELECT employer_details_func FROM dual;

3) In a PL/SQL Statements like,

dbms_output.put_line(employer_details_func);

This line displays the value returned by the function.

Activity to be Submitted by Students

Problem Statement :

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 and900 category is first class, if marks 899 and 825 category is Higher Second Class.

CONCLUSION:

Students are able to PL/SQL Stored Procedure and Stored Function

FAQ :

1. What are the features of PL/SQL?
2. What do you understand by PL/SQL table?
3. Explain the basic structure followed in PL/SQL?
4. When does a DECLARE block become mandatory?
5. How do you write comments in a PL/SQL code?
6. Can you explain the PL/SQL execution architecture?
7. Why is SYSDATE and USER keywords used?
8. Differentiate between SQL and PL/SQL.
9. What is the importance of %TYPE and %ROWTYPE data types in PL/SQL?
10. What are the various functions available for manipulating the character data?

ASSIGNMENT NO .6

AIM: 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_Roll Call 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.
Note: Instructor will frame the problem statement for writing PL/SQL block using all types of Cursors in line with above statement

COURSE OBJECTIVE:
Implement PL/SQL Code block for given requirements

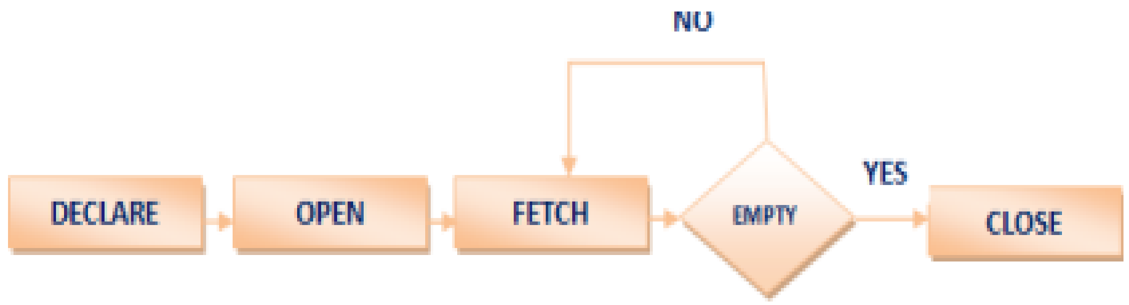
COURSE OUTCOME:

Implement PL/SQL Code block for given requirements

ENVIRONMENT: - Mysql

THEORY:
PL/SQL Cursor

When you work with database, you work with a complete set of rows returned from an SQL SELECT statement. However the application in some cases cannot work effectively with the entire result set, therefore, the database server needs to provide a mechanism for the application to work with one row or a subset of the result set at a time. As the result, Oracle created PL/SQL cursor to provide these extensions.
A PL/SQL cursor is a pointer that points to the result set of an SQL query against database tables. Working with PL/SQL Cursor
The following picture describes steps that you need to follow when you work with a PL/SQL cursor:



Declaring PL/SQL Cursor
To use PL/SQL cursor, first you must declare it in the declaration section of PL/SQL block or in a package as follows:

- First, you declare the name of the cursor after the keyword. The name of the cursor can have up to 30 characters in length and follows the naming rules of identifiers in PL/SQL. It is important to note that cursor's name is not a variable so you cannot use it as a variable such as assigning it to other cursor or using it in an expression. The parameter1,parameter2… are optional elements in the cursor declaration. These parameters allow you to pass arguments into the cursor. The is also an optional part.

- Second, you specify a valid SQL statement that returns a result set where the cursor points to.
- Third, you can indicate a list of columns that you want to update after the FOR UPDATE OF. This part is optional so you can omit it in the CURSOR declaration.

PL/SQL Cursor Attributes
These are the main attributes of a PL/SQL cursor and their descriptions.

Attribute	**Description**
cursor_name%FOUND	returns TRUE if record was fetched successfully by cursor cursor_name
cursor_name%NOTFOUND	returns TRUE if record was not fetched successfully by cursor cursor_name
cursor_name%ROWCOUNT	returns the number of records fetched from the cursor cursor_name at the time we test
cursor_name%ISOPEN	returns TRUE if the cursor cursor_name is open

Explicit Cursors

An explicit cursor is defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row. We can provide a suitable name for the cursor.
General Syntax for creatinga cursor is as given below:

CURSOR cursor_name IS select_statement;

Here,
cursor name – A suitable name for the cursor.
select_statement – A select query which returns multiple rows.

General Syntax to open a cursor is:
OPEN cursor_name;

General Syntax to fetch records from a cursor is:
FETCH cursor_name INTO record_name;
Or
FETCH cursor_name INTO variable_list;

General Syntax to close a cursor is:
CLOSE cursor_name;

When a cursor is opened, the first row becomes the current row. When the data is fetched it is copied to the record or variables and the logical pointer moves to the next row and it becomes the current row. On every fetch statement, the pointer moves to the next row. If you want to fetch after the last row, the program will throw an error. When there is more than one row in a cursor we can use loops along with explicit cursor attributes to fetch all the records.

FOR LOOP
CursorSyntax :

The syntax for the CURSOR FOR LOOP in Oracle/PLSQL is:

```
FOR record_index in cursor_name  LOOP
{
...statements...

}END LOOP;
```

Parameters or Arguments record_index: The index of the record.
cursor_name: The name of the cursor that you wish to fetch records from.
statements: The statements of code to execute each pass through the CURSOR FOR LOOP.

Parameterized cursor:

PL/SQL Parameterized cursor pass the parameters into a cursor and use them in to query. PL/SQL Parameterized cursor define only datatype of parameter and no need to define its length. Default values are assigned to the Cursor parameters and scope of the parameters are local.
Parameterized cursors are also saying static cursors that can pass parameter value when cursors are opened. Following example introduce the parameterized cursor.

emp_information table =>

EMP_NO	EMP_NAME
1	Forbs ross
2	marks jems
3	Saulin
4	Zenia Sroll

Example Code

Cursor display employee information from emp_information table whose emp_no four (4).

```
 DECLARE
cursor c (no number) is
select * from emp_information where emp_no = no;
tmp emp_information%rowtype;
BEGIN
OPEN c(4);

FOR tmp IN c(4) LOOP
dbms_output.put_line('EMP_No:'||tmp.emp_no);
dbms_output.put_line('EMP_Name: '||tmp.emp_name);
END Loop;
CLOSE
c;
END;
 /
SQL>@parameter_cursor_demo
EMP_No: 4
EMP_Name: Zenia
```

Activity to be Submitted by Students

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.

CONCLUSION: We have implemented all types of Cursors successfully.

FAQ:

1. What is a PL/SQL cursor?
2. What is the use of WHERE CURRENT OF in cursors?
3. How can a name be assigned to an unnamed PL/SQL Exception Block?
4. What is the purpose of WHEN clause in the trigger?
5. Differentiate between implicit cursor and explicit cursor.

ASSIGNMENT NO .7

AIM: Database Trigger (All Types: Row level and Statement level triggers, Before and AfterTriggers).

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.
Note: Instructor will Frame the problem statement for writing PL/SQLblock for all types of Triggers in line with above statement.

COURSE OBJECTIVE:

Implement PL/SQL Code block for given requirements

COURSE OUTCOME:

Implement PL/SQL Code block for given requirements

ENVIRONMENT: **-** Mysql

THEORY:

PL/SQL Trigger
Trigger is invoked by Oracle engine automatically whenever a specified event occurs.Trigger is stored intodatabase and invoked repeatedly, when specific condition match.
Triggers are stored programs, which are automatically executed or fired when some event occurs.Triggers are 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 could be defined on the table, view, schema, or database with which the event is associated. Advantages of Triggers
These are the following advantages of Triggers:

- Trigger generates some derived column values automatically
- Enforces referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions Creating a trigger:

Syntax for creating trigger:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col_name]

ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
Declaration-statements
BEGIN
Executable-statements
EXCEPTION
Exception-handling-statements
END;
```

Here,

- CREATE [OR REPLACE] TRIGGER trigger_name:
  It creates or replaces an existing trigger with the trigger_name.
- {BEFORE | AFTER | INSTEAD OF} :
  This specifies when the trigger would be executed.
- The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE}:
  This specifies the DML operation.
- [OF col_name]:
  This specifies the column name that would be updated.
- [ON table_name]:
  This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n]:
  This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
- [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.
- WHEN (condition):
  This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Example:

The price of a product changes constantly. It is important to maintain the history of the prices of the products. We can create a trigger to update the 'product_price_history' table when the price of the product is updated in the 'product' table.

1) Create the 'product' table and 'product_price_history' table as follows:

```
CREATE TABLE product_price_history (product_id number(5), product_name varchar2(32),
supplier_name varchar2(32), unit_price number(7,2) );


CREATE TABLE product (product_id number(5), product_name varchar2(32), supplier_name varchar2(32),
unit_price number(7,2) );
```

2) Create the price_history_trigger and execute it.

```
 CREATE or REPLACE TRIGGER price_history_trigger BEFORE UPDATE OF unit_price
ON product

FOR EACH ROW
BEGIN
INSERT INTO product_price_history VALUES

(:old.product_id,

:old.product_name,

:old.supplier_name,

:old.unit_price);

END;

/
```

3) Lets update the price of a product.

```
UPDATE PRODUCT SET unit_price = 800 WHERE product_id = 100
```

Once the above update query is executed, the trigger fires and updates the 'product_price_history' table.

4) If you ROLLBACK the transaction before committing to the database, the data inserted to the table is also rolled back.

Types of PL/SQL Triggers

There are two types of triggers based on the which level it is triggered.

1) Row level trigger - An event is triggered for each row upated, inserted or deleted.
2) Statement level trigger - An event is triggered for each sql statement executed. PL/SQL Trigger Execution Hierarchy

The following hierarchy is followed when a trigger is fired.

1) BEFORE statement trigger fires first.
2) Next BEFORE row level trigger fires, once for each row affected.
3) Then AFTER row level trigger fires once for each affected row. This events will alternates between BEFORE and AFTER row level triggers.
4) Finally the AFTER statement level trigger fires.

For Example: Let's create a table 'product_check' which we can use to store messages when triggers are fired.

```
CREATE TABLE product (Message varchar2(50), Current_Date number(32) );
```

Let's create a BEFORE and AFTER statement and row level triggers for the product table.

1) BEFORE UPDATE, Statement Level:

This trigger will insert a record into the table 'product_check' before a sql update statement is executed, at the statement level.

```
CREATE or REPLACE TRIGGER Before_Update_Stat_product BEFORE

UPDATE ON product Begin
INSERT INTO product_check Values('Before update, statement level',sysdate);
END;
/
```

2) BEFORE UPDATE, Row Level:

This trigger will insert a record into the table 'product_check' before each row is updated.

```
CREATE or REPLACE TRIGGER Before_Upddate_Row_product BEFORE
UPDATE ON product FOR EACH ROW
BEGIN
INSERT INTO product_check Values('Before update row level',sysdate); END;
/
```

3) AFTER UPDATE, Statement Level:

This trigger will insert a record into the table 'product_check' after a sql update statement is executed, at the statement level.

```
CREATE or REPLACE TRIGGER After_Update_Stat_product AFTER
UPDATE ON product
BEGIN
INSERT INTO product_check Values('After update, statement level', sysdate);
End;
/
```

4) AFTER UPDATE, Row Level:

This trigger will insert a record into the table 'product_check' after each row is updated.

```
CREATE or REPLACE TRIGGER After_Update_Row_product
 AFTER  insert On product
 FOR EACH ROW
  BEGIN
INSERT INTO product_check Values('After update, Row level',sysdate);
END;

/
```

Now lets execute a update statement on table product.

```
UPDATE PRODUCT SET unit_price = 800 WHERE product_id in (100,101);
```

Lets check the data in 'product_check' table to see the order in which the trigger is fired.

```
SELECT * FROM product_check;
```

Output:

```
Mesage Current_Date

Before update, statement level    26-Nov-2008
Before update, row level          26-Nov-2008
After update, Row level           26-Nov-2008
Before update, row level          26-Nov-2008
After update, Row level           26-Nov- 2008
After update, statement level      26-Nov-2008
```

Activity to be Submitted by Students

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.

CONCLUSION: We have implemented all types of Triggers successfully.

FAQ:

Q1 : What is a Trigger?
Ans: Trigger is a named PL/SQL block stored as a stand-alone object in an Oracle database and is implicitly executed when the triggering event occurs.

You can not make a call to a Trigger to fire. They are never executed unless the triggering event occurs.

Q2 : What is a triggering event?
Ans: Triggering event is a database operation associated with either a table, a view, a schema, or the database. The database operations could be,

DML statement
DDL statement
Database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN)

Q3 : Types of trigger based on triggering event?
Ans: Based on the database operation there could be,

DML trigger- Fired on INSERT, UPDATE, DELETE
DDL trigger- Fired on CREATE, ALTER, DROP, TRUNCATE, RENAME
System trigger- Fired on SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN

Q4 : What are some use cases of Triggers??
Ans: Triggers are commonly used to,

- Automatically generate derived column values
- Prevent invalid transactions
- Enforce complex security authorizations
- Enforce complex business rules
- Provide sophisticated auditing
- Maintain synchronous table replicates

Q6 : Difference between Triggers and Stored procedures?
Ans: Procedures and triggers differ in the way, they are invoked. A stored procedure is executed explicitly by issuing procedure call statement from another block.

Triggers are implicitly fired (executed) by Oracle when a triggering INSERT, UPDATE, or DELETE statement is issued on the associated table, no matter which user is connected or which application is being used.
A trigger can include SQL and PL/SQL statements and can invoke stored procedures. However Stored procedures do not call triggers directly. Stored procedures include DML statements that can invoke triggers.
Procedures may have parameters, whereas triggers are not parameterized.

Q7 : What is the maximum number of triggers, you can apply on a single table?
Ans: Triggers in Oracle has following parameters and can be created with any combination of-

- 2 types based on triggering time – Before or After
- 2 types based on trigger type – Statement or Row level
- 3 types based on event type – Insert or Update or Delete

Hence, 2*2*3 = 12 types of triggers are possible with all combination.

ASSIGNMENT NO .8

AIM:

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.)

OBJECTIVES:
Implement PL/SQL Code block for given requirements

OUTCOMES:
C306.6
Design and develop application considering actual requirements and using database concepts

ENVIRONMENT:
Mysql/eclipse/ wampserver/php/oracle

THEORY:
Why use JDBC
Before JDBC, ODBC API was the database API to connect and execute query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).
JDBC Driver
1. JDBC Drivers
1. JDBC-ODBC bridge driver
2. Native-API driver
3. Network Protocol driver
4. Thin driver

JDBC Driver is a software component that enables java application to interact with the database.There are 4 types of JDBC drivers:
1. JDBC-ODBC bridge driver

2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

1) JDBC-ODBC bridge driver
The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.
Advantages:
- easy to use.
- can be easily connected to any database. Disadvantages:
- Performance degraded because JDBC method call is converted into the ODBC function calls.
- The ODBC driver needs to be installed on the client machine.

2) Native-API driver
The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.
Advantage:
 performance upgraded than JDBC-ODBC bridge driver. Disadvantage:
 The Native driver needs to be installed on the each client machine

The Vendor client library needs to be installed on client machine.
3) Network Protocol driver
The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.
Advantage:
 No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

Disadvantages:
 Network support is required on client machine.
 Requires database-specific coding to be done in the middle tier.
 Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

4) Thin driver
The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.
Advantage:
 Better performance than all other drivers.
 No software is required at client side or server side. Disadvantage:
 Drivers depends on the Database.
5 Steps to connect to the database in java They are as follows:
 Register the driver class
 Creating connection
 Creating statement
 Executing queries
 Closing connection

1) Register the driver class
The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.
Syntax of forName() method
1. public static void forName(String className)throws ClassNotFoundException

Example to register the OracleDriver class
1. Class.forName(&quot;oracle.jdbc.driver.OracleDriver&quot;);
2) Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database.
Syntax of getConnection() method

3) public static Connection getConnection(String url)throws SQLException
4) public static Connection getConnection(String url,String name,String password) throws SQLException
Example to establish connection with the Oracle database
5) Connection con=DriverManager.getConnection(
6) &quot;jdbc:oracle:thin:@localhost:1521:xe&quot;,&quot;system&quot;,&quot;password&quot;);
7) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.
Syntax of createStatement() method
public Statement createStatement()throws SQLException
Example to create the statement object
Statement stmt=con.createStatement();
8) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.
Syntax of executeQuery() method
public ResultSet executeQuery(String sql)throws SQLException
Example to execute query
ResultSet rs=stmt.executeQuery(&quot;select * from emp&quot;);
while(rs.next()){ System.out.println(rs.getInt(1)+&quot; &quot;+rs.getString(2));
}
9) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.
Syntax of close() method
public void close()throws SQLException Example to close connection con.close();
import java.sql.*; class OracleCon{
public static void main(String args[]){ try{
Class.forName(&quot;oracle.jdbc.driver.OracleDriver&quot;);
Connection con=DriverManager.getConnection(
“jdbc:oracle:thin:@localhost:1521:xe&quot;,&quot;system&quot;,&quot;oracle&quot;);
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery(&quot;select * from emp&quot;);
while(rs.next())
System.out.println(rs.getInt(1)+&quot; &quot;+rs.getString(2)+&quot; &quot;+rs.getString(3));
con.close();
}catch(Exception e)
{ System.out.println(e);} } }
PHP Connectivity
PHP will require that mysql is enabled (it is on most PHP set ups).
$db =
mysqli_connect(&#39;localhost&#39;,&#39;username&#39;,&#39;password&#39;,&#39;database_name&#39;)
or die(&#39;Error connecting
to MySQL server.&#39;);
The variable $db is created and assigned as the connection string, it will be used in future steps.

If there is a failure then an error message will be displayed on the page. If it is successful you will see PHP connect to MySQL.
Performing a database query
The mysql query is actually performed in the body of the html page, so additional php opening and closing tags will be required. For the query we are going to specify a read of all fields from a given table. The $query variable selects all rows in the table. You just need to use your table name.

```
&lt;?php
$query = &quot;SELECT * FROM table_name&quot;;

mysqli_query($db, $query) or die(&#39;Error querying database.&#39;);
```

Again the returned page in the browser should be blank and error free, if you do receive the error – 'Error querying database..' check the table name is correct.

```
&lt;?php
//Step1
$query = &quot;SELECT * FROM table_name&quot;;
mysqli_query($db, $query) or die(&#39;Error querying database.&#39;);
$result = mysqli_query($db, $query);
$row = mysqli_fetch_array($result);
while ($row = mysqli_fetch_array($result)) {
echo $row[&#39;first_name&#39;] . &#39; &#39; . $row[&#39;last_name&#39;] . &#39;: &#39; .
$row[&#39;email&#39;] . &#39; &#39; . $row[&#39;city&#39;] .&#39;&lt;br /&gt;&#39;;}?&gt;
&lt;/body&gt;
&lt;/html&gt;
```

Closing off the connection
Closing the connection will require another set off opening and closing php tags after the closing html tag.

```
&lt;?php
//Step1
$db = mysqli_connect(&#39;localhost&#39;,&#39;root&#39;,&#39;root&#39;,&#39;database_name&#39;) or
die(&#39;Error connecting to
MySQL server.&#39;);
?&gt;
&lt;html&gt;
&lt;head&gt;
&lt;/head&gt;
&lt;body&gt;
&lt;h1&gt;PHP connect to MySQL&lt;/h1&gt;
&lt;?php
//Step2
$query = &quot;SELECT * FROM table_name&quot;;
mysqli_query($db, $query) or die(&#39;Error querying database.&#39;);
//Step3
$result = mysqli_query($db, $query);

$row = mysqli_fetch_array($result);
while ($row = mysqli_fetch_array($result))
{
echo $row[&#39;first_name&#39;] . &#39; &#39; . $row[&#39;last_name&#39;] . &#39;: &#39; .
$row[&#39;email&#39;] . &#39; &#39; . $row[&#39;city&#39;] .&#39;&lt;br /&gt;&#39;;
}
//Step 4 mysqli_close($db);
?&gt;
&lt;/body&gt;
&lt;/html&gt;
```

Activity to be Submitted by Students
1. Login page in PHP

Conclusion: We have implemented database connectivity

FAQ:
1. What is JDBC?
JDBC is a Java API that is used to connect and execute the query to the database. JDBC API uses JDBC drivers to connect to the database. JDBC API can be used to access tabular data stored into any relational database.

2. What is JDBC Driver?
3. What are the different steps for database connectivity?
4. What are the JDBC API components?
5. What are the JDBC statements?

ASSIGNMENT NO .9

AIM:

MongoDB Queries

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

OBJECTIVES:

To acquire the skills to use a powerful, flexible, and scalable general-purpose databases to handle Big Data .

OUTCOMES:

Implement NoSQL queries using MongoDB

ENVIRONMENT:

Mongodb

THEORY:

The use Command use DATABASE_NAME

use DATABASE_NAME is used to create database. The command will create a new database, if it doesn't exist otherwise it will return the existing database.

Syntax:

Example: If you want to create a database with name **<mydb>**, then **use DATABASE** statement would be as follows:

>use mydb

To check your currently selected database use the command **db**

>db;

The dropData

db.dropDatabase() command is used to drop a existing database.

Syntax: Basic syntax of **dropDatabase()** command is as follows:

db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database

Create Collection The createCollection() Method

db.createCollection(name, options) is used to create collection. db.createCollection(name, options)

Syntax:

In the command, **name** is name of collection to be created. **Options** is a document and used to specify configuration of collection

The drop() Method

db.collection.drop() is used to drop a collection from the database. db.COLLECTION_NAME.drop()

Syntax:

Insert Document The insert() Method
To insert data into MongoDB collection, you need to use MongoDB's ***insert()*** *or* ***save()*** method.
Syntax:
Basic syntax of **insert()** command is as follows:
>db.COLLECTION_NAME.insert(document)
Example
>db.stud1.insert({name:’abc’,age:20})

Query Documents (Find)
In MongoDB, the db.collection.find() method retrieves documents from a collection. The db.collection.find() method returns a *cursor* to the retrieved documents. The db.collection.findOne() method also performs a read operation to return a single document.
1. **Select All Documents in a Collection**
An empty query document ({}) selects all documents in the collection:
db.inventory.find({})
Not specifying a query document to the find() is equivalent to specifying an empty query document. Therefore the following operation is equivalent to the previous operation:
db.inventory.find()

Specify Equality Condition
To specify equality condition, use the query document { <field>: <value> } to select all documents that contain the <field> with the specified <value>. The following example retrieves from the inventory collection all documents where the type field has the value snacks:
db.inventory.find({ type: "snacks" })

Specify Conditions Using Query Operators
A query document can use the *query operators* to specify conditions in a MongoDB query.
The following example selects all documents in the inventory collection where the value of the type field is either 'food' or 'snacks':
db.inventory.find({ type: { $in: ['food', 'snacks'] } })
Although you can express this query using the $or operator, use the $in operator rather than the $or operator when performing equality checks on the same field.

Comparison Operators and Logical Operators : For comparison of different BSON type values, see the specified BSON comparison order.

Comparison Operator Name Description
$eq Matches values that are equal to a specified value.
$gt Matches values that are greater than a specified value.
$gte Matches values that are greater than or equal to a specified value.
$in Matches any of the values specified in an array.
$lt Matches values that are less than a specified value.
$lte Matches values that are less than or equal to a specified value.

$ne Matches all values that are not equal to a specified value.
$nin Matches none of the values specified in an array.

Logical Operators Name Description
$and Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not Inverts the effect of a query expression and returns documents that do not match the query expression.
$nor Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
$or Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

SAVE METHOD
The save() method has the following form:
db.collection.save(<document>, { writeConcern: <document> })

The products collection contains the following document:
{ "_id" : 100, "item" : "water", "qty" : 30 }

The save() method performs an update with upsert:true since the document contains an _id field:
db.products.save({ _id : 100, item : "juice" })

MongoDB Logical Query Operator - $and & $not
Logical Operator - $and
The MongoDB $and operator performs a logical AND operation on an array of two or more expressions and retrieves the documents which satisfy all the expressions in the array. The $and operator uses short-circuit evaluation. If the first expression (e.g. <expression1>) evaluates to false, MongoDB will not evaluate the remaining expressions.
Syntax:
{ $and: [{ <exp1> }, { <exp2> } , ... , { <expN> }] }

Our database name is 'myinfo' and our collection name is 'student'. Here, is the collection bellow. Example of MongoDB Logical Operator - $and

If we want to select all documents from the collection "student" which satisfying the condition -
1. sex of student is Female and

2. class of the student is VI and

3. grd_point of the student is greater than equal to 31 the following mongodb command can be used :

```
>db.student.find({$and:[{"sex":"Male"},{"grd_point":{ $gte: 31 }},{"class":"VI"}]}).pretty();
```

Logical Operator - $not
The MongoDB $not operator performs a logical NOT operation on the given expression and fetches selected documents that do not match the expression and the document that do not contain the field as well, specified in the expression.

Syntax:
{ field: { $not: { <expression> } } }

Example of MongoDB Logical Operator –

$not
If we want to select all documents from the collection "student" which satisfying the condition -age of the student is at least 12 the following mongodb command can be used :

>db.student.find({"age": { $not: {$lt : 12}}}).

pretty();
The following mongodb command can be used :
>db.student.find({"sex": { $not: /^M.*/}}).
pretty();

Activity to be Submitted by Students
1. Collection “orderinfo“ which contains the documents given as below(Perform on Mongo Terminal)
{
cust_id:123
cust_name:”abc”,
status:”A”,
price:250
}
i. find the average price for each customers having status 'A'
ii. Display the status of the customers whose amount/price lie between 100 and 1000
iii. Display the customers information without “_id” .
iv. create a simple index on onderinfo collection and fire the queries.
2. Collection “movies“ which contains the documents given as below(Perform on Mongo Terminal)
{
name: “Movie1”,
type: “action”,
budget:1000000
producer:{
name: “producer1”,
address:”PUNE”
} }
i. Find the name of the movie having budget greater than 1,00,000.
ii. Find the name of producer who lives in Pune
iii. Update the type of movie “action” to “horror”
iv. Find all the documents produced by name “producer1” with their address

Conclusion: We have implemented CRUD operations using Mongodb

FAQ:

1. What is MongoDB?
2. How MongoDB works?
3. Difference between RDBMS and MongoDB.
4. What are the data types used in MongoDB?
5. Create Spring Boot – CRUD operations using MongoDB.

ASSIGNMENT NO .10

AIM:

MongoDB – Aggregation and Indexing:

Design and Develop MongoDB Queries using aggregation and indexing with suitable example using MongoDB.

OBJECTIVES:

To acquire the skills to use a powerful, flexible, and scalable general-purpose databases to handle Big Data

OUTCOMES:

Implement NoSQL queries using MongoDB

ENVIRONMENT:

Mongodb

THEORY:

INDEXING

Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a collection to select those documents that match the query statement. This scan is highly inefficient and requires the **mongod** to process a large volume of data. “Indexes are special data structures that store a small portion of the data set in an easy to traverse form.”

The index stores the value of a specific field or set of fields, ordered by the value of the field as specified in index.

The ensureIndex() Method

To create an index you need to use ensureIndex() method of mongodb.

SYNTAX:

Basic syntax of **ensureIndex()** method is as follows()

```
>db.COLLECTION_NAME.ensureIndex({KEY:1})
```

Here key is the name of filed on which you want to create index and 1 is for ascending order. To create index in descending order you need to use -1.

EXAMPLE

```
>db.mycol.ensureIndex({"title":1})
```

In **ensureIndex()** method you can pass multiple fields, to create index on multiple fields.

```
>db.mycol.ensureIndex({"title":1,"description":-1})
```

Drop Index :

```
>db.mycol.dropIndex({"title":1,"description":-1})
```

AGGREGATION :

Aggregations operations process data records and return computed results. operations on the grouped data to return a single result. In sql count(*) and with group by is an equivalent of MongoDB
aggregation The aggregate() Method For the aggregation in mongodb you should use aggregate() method.SYNTAX: Basic syntax of aggregate() method is as follows
Now from the above collection if you want to display a list that how many tutorials are

```
{
_id: ObjectId(7df78ad8902c) title: 'MongoDB Overview',
description: 'MongoDB is no sql database', by_user:

'tutorials point',

url: 'http://www.tutorialspoint.com', tags:
['mongodb', 'database', 'NoSQL'], likes: 100},
{_id: ObjectId(7df78ad8902d) title: 'NoSQL Overview',
description: 'No sql database is very fast', by_user:

'tutorials point',

url: 'http://www.tutorialspoint.com', tags:
 ['mongodb', 'database', 'NoSQL'], likes: 10
}
}
```

written by each user then you will use **aggregate()** method as shown below:

```
> db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : 1}}}])
{"result" : [{"_id" : "tutorials point", "num_tutorial" : 2},
{"_id" : "tutorials point", "num_tutorial" : 1}],"ok" : 1}
```

SQL equivalent query for the above use case will be **select by_user, count(*) from mycol group by by_user.** In the above example we have grouped documents by field **by_user** and on each occurance of by_user previous value of sum is incremented. There is a list available aggregation expression.

Expression	**Description**	**Example**
$sum	Sumsup the defined valuedocuments in the from collection.	db.mycol.aggregate([{$group : {_id : "$by_user" num_tutorial : {$sum : "$likes"}}}])
$avg	Calculates the average of all given values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user" num_tutorial : {$avg : "$likes"}}}])

$min	Gets the minimum of the corresponding all values from documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
$max	Gets the maximum of the corresponding all values fromdocuments in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$push	Inserts the value to an array in the resulting document.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$addToSet	Inserts the value to an array in the resultingdocument but does not create duplicates.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])
$first	Gets the first document from the source documents according to the grouping.	db.mycol.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	Gets the last document from the source documents according to the grouping.	db.mycol.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

Pipeline Concept
In UNIX command shell pipeline means the possibility to execute an operation on some input and use the output as the input for the next command and so on. MongoDB also support same concept in aggregation framework. There is a set of possible stages and each of those is taken a set of documents as an input and is producing a resulting set of documents (or the final resulting JSON document at the end of the pipeline). This can then in turn again be used for the next stage an so on.
Possible stages in aggregation framework are following:
$project: Used to select some specific fields from a collection.

$match: This is a filtering operation and thus this can reduce the amount of documents that are given as input to the next stage.

$group: This does the actual aggregation as discussed above.

$sort: Sorts the documents.

$skip: With this it is possible to skip forward in the list of documents for a given amount of documents.

$limit: This limits the amount of documents to look at by the given number starting from the current position.s

$unwind: This is used to unwind document that are using arrays. when using an array the data is kind of pre-joined and this operation will be undone with this to have individual documents again. Thus with this stage we will increase the amount of documents for the next stage.

Activity to be Submitted by Students
Perform aggregation and Indexing using mongodb on below database
1. Create a database department
2. Create a collection as teacher with fields as name , department ,experience and salary
3. Display the department wise average salary.
4. Display the no. Of employees working in each department.
5. Display the department wise minimum salary.
6. Apply index and drop index

Conclusion: We have implemented aggregation and indexing using Mongodb

FAQ:
1. What is the use of aggregation and indexing in MongoDB?
2. What is Aggregation pipeline stage in MongoDB?
3. $substrCP(aggregation) operator in MongoDB?
4. What is the difference between MongoDB and Mysql?
5. How to create,define and drop a MongoDB collection?

ASSIGNMENT NO .11

AIM:

MongoDB – Map-reduces operations:

Implement Map reduces operation with suitable example using MongoDB.

OBJECTIVES:

To acquire the skills to use a powerful, flexible, and scalable general-purpose databases to handle Big Data

OUTCOMES:

Implement NoSQL queries using MongoDB

ENVIRONMENT:

Mongodb

THEORY:

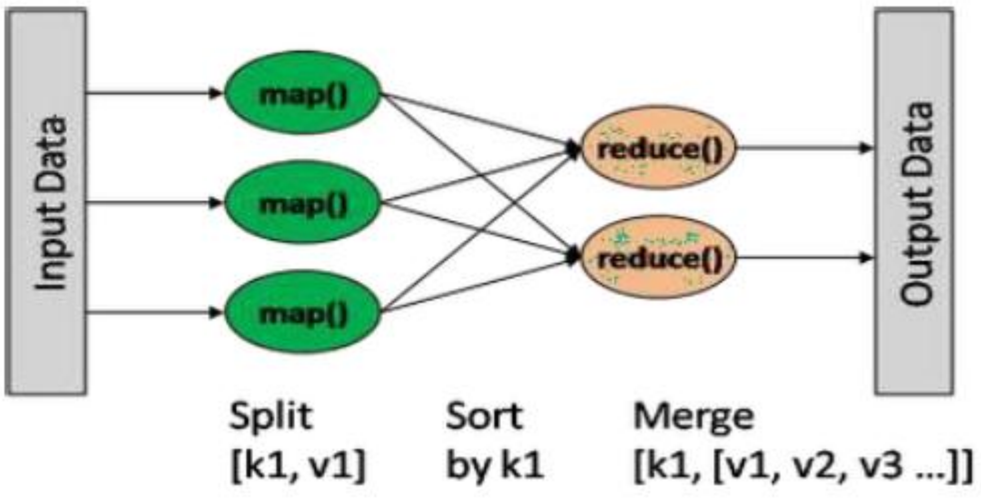
Map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results. MongoDB uses **mapReduce** command for map-reduce operations. MapReduce is generally used for processing large data sets. In simple terms, the mapReduce command takes 2 primary inputs, the mapper function and the reducer function .

Working of Mapper and Reducer Function :

MapReduce is a two-step approach to data processing. First you map, and then you reduce. The mapping step transforms the inputted documents and emits a key=>value pair (the key and/or value can be complex). Then, key/value pairs are grouped by key, such that values for the same key end up in an array. The reduce gets a key and the array of values emitted for that key, and produces the final result. The map and reduce functions are written in JavaScript. A Mapper will start off by reading a collection of data and building a Map with only the required fields we wish to process and group them into one array based on the key. And then this key value pair is fed into a Reducer, which will process the values.

MapReduce Command:

syntax of the basic mapReduce command: db.collection.mapReduce(function() {emit(key,value);}, //map function function(key,values) {return reduceFunction}, //reduce function
{out: collection, query: document, sort: document, limit: number})

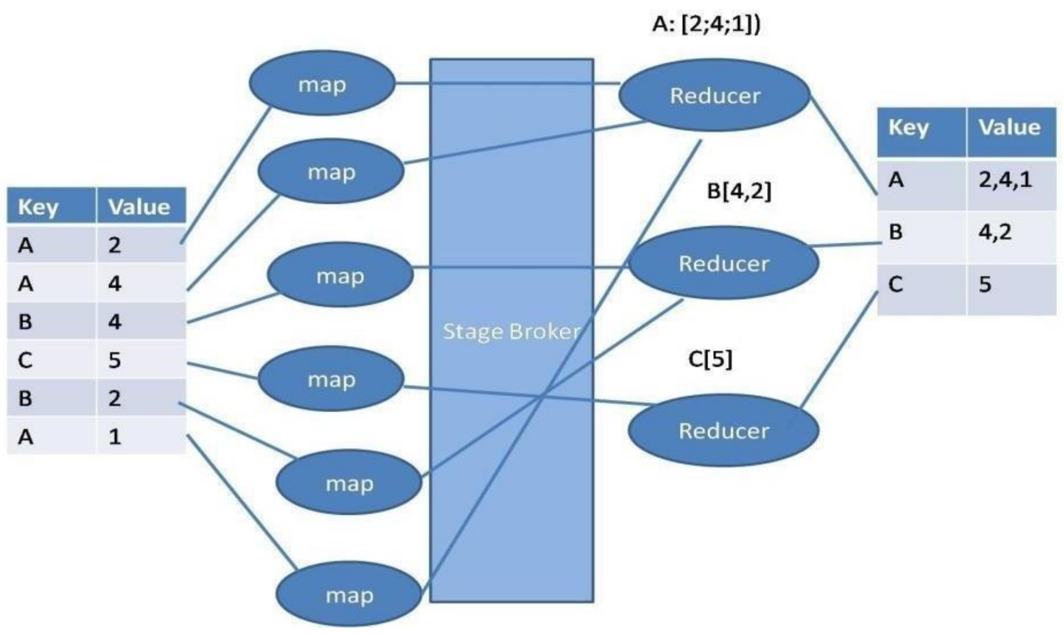


The map-reduce function first queries the collection, then maps the result documents to emit key-value pairs which is then reduced based on the keys that have multiple values. MapReduce Command:
syntax of the basic mapReduce command:

```
db.collection.mapReduce (function() {emit(key,value);}, //map function
function(key,values) {return reduceFunction}, //reduce function
{out: collection, query: document, sort: document, limit: number})
```

The map-reduce function first queries the collection, then maps the result documents to emit key-value pairs which is then reduced based on the keys that have multiple values.
In the above syntax:
map is a javascript function that maps a value with a key and emits a key-value pair
reduce is a javascript function that reduces or groups all the documents having the same key
out specifies the location of the map-reduce query result
query specifies the optional selection criteria for selecting documents
sort specifies the optional sort criteria
limit specifies the optional maximum number of documents to be returned
Map Reduce Example
The below example is to retrieve the sum of total values related to particular key.
1. Insert data in *mapCollection.*

```
db.mapc.insert({key:"a", value:2})
db.mapc.insert({key:"a", value:4})
```



Activity to be Submitted by Students

Collection "city " which contains the documents given as below(Perform on Mongo Terminal)

{
city:"pune",
type:"urban",
state:"MH",
population:"5600000"
}

-using mapreduce, find statewise population
-using mapreduce, find citywise population
-using mapreduce, find typewise

CONCLUSION: We have implemented Map reduce using Mongodb Successfully

FAQ:

1. Create database command used for Map reduce function.
2. MongoDB applies which phase to input document in Map reduce.
   The Map function emits key value pairs, hence Map phase applies.
3. What are the purposes of Map reduce in MongoDB?
4. What are the methods used in MongoDB?
5. How to implement MongoDB Map Reduce using Aggregation?

ASSIGNMENT NO .12

AIM:

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.)

OBJECTIVES:

Implement PL/SQL Code block for given requirements

OUTCOMES:

C306.6

Design and develop application considering actual requirements and using database concepts

ENVIRONMENT:

Mongodb/eclipse/ wampserver/php

Mongodb Connectivity Mongo connection

Connect to MongoDB server.

For MongoDB version >= 2.10.0, uses MongoClient.

```
// Old version, uses Mongo
Mongo mongo = new Mongo("localhost", 27017);
// Since 2.10.0, uses MongoClient
MongoClient mongo = new MongoClient( "localhost" , 27017 );
```

If MongoDB in secure mode, authentication is required.

```
MongoClient mongoClient = new MongoClient();
DB db = mongoClient.getDB("database name");
boolean auth = db.authenticate("username", "password".toCharArray());
```

Mongo Database

Get database. If the database doesn’t exist, MongoDB will create it for you.

```
DB db = mongo.getDB("database name");
```

Display all collections from selected database.

```
DB db = mongo.getDB("testdb");
```

Save example

Save a document (data) into a collection (table) named “user”.

```
DBCollection table = db.getCollection("user");
BasicDBObject document = new BasicDBObject();
 document.put("name", "mkyong");
document.put("age", 30);
document.put("createdDate", new Date());
 table.insert(document);
```

Update example

Update a document where “name=mkyong”.

```
DBCollection table = db.getCollection("user");
BasicDBObject query = new BasicDBObject();
query.put("name", "mkyong");
BasicDBObject newDocument = new BasicDBObject();
newDocument.put("name", "mkyong-updated");
BasicDBObject updateObj = new BasicDBObject();
updateObj.put("$set", newDocument);
table.update(query, updateObj);
```

Find example

Find document where “name=mkyong”, and display it with DBCursor

```
DBCollection table = db.getCollection("user");
BasicDBObject searchQuery = new BasicDBObject();
 searchQuery.put("name", "mkyong");
DBCursor cursor = table.find(searchQuery);
while (cursor.hasNext())
 {
System.out.println(cursor.next());
}
```

Delete example

Find document where “name=mkyong”, and delete it.

```
DBCollection table = db.getCollection("user");
BasicDBObject searchQuery = new BasicDBObject();
searchQuery.put("name", "mkyong");
 table.remove(searchQuery);
```

Activity to be Submitted by Students

1. CRUD operation using Mongodb and JDBC connectivity

Conclusion: We have implemented database connectivity

FAQ:

1. How to access data from MongoDB database?

To select data from a table in MongoDB, we can also use the find() method. The find() method returns all occurrences in the selection. The first parameter of the find() method is a query object. In this example we use an empty query object, which selects all documents in the collection.

2. What are some of the advantages of MongoDB?

3. What is a Document in MongoDB?

4. What is a Collection in MongoDB?

5. What are Databases in MongoDB?

6. What is the Mongo Shell?

7. How does Scale-Out occur in MongoDB?

8. What are some features of MongoDB?

9. How to add data in MongoDB?