2078
V 0.1

Lab Manual

on

Database Management System

for

BCT, BEI

Department of Electronics and Computer Engineering

Compiled By:

Rama Bastola

Lecturer

Department of Electronics and Computer Engineering

IOE, Thapathali Campus

Table of Contents

Introduction

A database is a repository of logically related and similar data. It is an organized collection of related information so that it can easily be accessed, managed and updated. For example, dictionary, university database, library. A university database might contain information about students, faculty, courses, and classrooms; relationships between entities, such as student's enrollment in courses, faculty teaching courses, and the use of rooms for courses.

A database management system, or DBMS is a collection of interrelated data and a set of programs to access those data which provides a way to store and retrieve database information that is both convenient and efficient. DBMS is designed to manage large bodies of information which involves defining structure for storage of information; providing mechanism for manipulation of information; data security; data integrity and consistency. Example: IBM's DB2, Microsoft's SQL Server, Oracle, Sybase.

DBMS is essential to every business such as Google, Yahoo!, Amazon.com, or thousands of organizations that provide information, there is a database behind the scenes serving up the information you request. Corporations maintain all their important records in databases. Databases are likewise found at the core of many scientific investigations.

RDBMS is acronym for Relation Database Management System. Dr. E. F. Codd first introduced the Relational Database Model in 1970. The Relational model allows data to be represented in a simple row- column. Each data field is considered as a column and each record is considered as a row. In relational model there is relation between their data elements. Data is stored in tables. Tables have columns, rows and names. Tables can be related to each other if each has a column with a common type of information. The most famous RDBMS packages are Oracle, MySQL

Student			
Sid	Sname	Sbatch	Sprogram
1	Ram Joshi	2075	BCT

Student_dtl			
Sid	Sphone	Semail	Saddress
1	9841414141	ram@gmail.com	Thapathali

In support of the relational model, Dr. EF-Codd also proposed 12 rules which are known as **12 golden rules of EF-Codd**.

Zero Rules: For a system, it must be able to qualify as an RDBMS and to manage its databases entirely through its relational capabilities.

Rule 1: Information Rule: The data stored in a database (maybe user data or metadata) must be a value of some table cell, everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule: Every single data element value has guaranteed to be accessible logically with a combination of table name, attribute name, primary key. No other means such as pointer can be used to access data.

Rule 3: Systematic treatment of null values: The null values in the database create an empty cell. This null value in a database must be given a systematic and uniform treatment. This is a very important rule because a null can be interpreted as one of the following: data is missing, data is not known, data is not applicable.

Rule 4: Active online catalog: The structure description of the entire database must be stored in an online catalog known as the data dictionary. Which can be accessed by the authorized user. Users can use the same query language to access the catalog which they use to access DB itself.

Rule 5: Comprehensive data sub-language rule: A database can be used only be access using language having linear syntax that supports data definition, data manipulation & transaction management operation. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considering as a violation.

Rule 6: View Updating Rule: All the views of the database which can theoretically be updated must also be updatable by the system.

Rule 7: High level insert, update, delete Rule: A DB must support high-level insertion, updation, and deletion. This must not the limited to a single row i.e. it must support union, intersection & (-) minus operation to yield sets of data records.

Rule 8: Physical data independence: The data stored in a database must be independent of the application that access the database. Any changes in the physical structure of the database must nor have any impact on how the data is being accessed by an external application.

Rule 9: Logical data independence: The logical data and database must be independent of its user view application any changes in logical data must not affect the application using it.

Rule 10: Integrating independence: A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application.

Rule 11: Distribution independence: The end user must not be able to see that the data is distributed at the over the various location. Users should always get the impression that the data is located one site only.

Rule 12: Non Sub-version rule: If a system has an interface that provides access to low-level records then the interface must not be able to bypass security and integrity constraints.

Database Languages

A DBMS must support one or more languages with which application developers can set up, populate, scrutinize, and maintain data. In principle, different purposes are best served by different database languages, but in practice, DBMSs tend to use one language to cater for more than one purpose.

i. Data Definition Language (DDL)
  - Used to specify a database schema
  - DDL statements are compiled, resulting in a set of tables stored in a special file called a **data dictionary or data directory.**
  - The data directory contains metadata (data about data)
  - **Basic idea:** hide implementation details of the database schemas from the users
  - **Example DDL Statements:**
    - **CREATE :**Creates a new database or object, such as a table, index or column
    - **ALTER:** Changes the structure of the database or object
    - **DROP:** Deletes the database or existing objects
    - **RENAME:** Renames the database or existing objects
  - DDL provides facilities to specify consistency constraints such as Domain Constraints, Referential Integrity

ii. **Data Manipulation Language (DML)**
DML is a language which enables users to **access and manipulate data**
Data Manipulation is:
  - **INSERT:** insertion of new information into the database

- **SELECT:** retrieval of information from the database
- **DELETE:** deletion of information in the database
- **UPDATE:** modification of information in the database

The goal is to provide efficient human interaction with the system.

Two types of DML

- **Procedural DMLs** require a user to specify *what* data are needed and *how* to get those data
- **Declarative DMLs** (**nonprocedural DMLs**) require a user to specify *what* data are needed *without* specifying how to get those data

iii. **Data Control Language (DCL)**

A Data Control Language (DCL) is used for controlling privilege in the database which allows database administrators to configure security access to databases Examples of DCL commands include:

- **GRANT**: allow specified users to perform specified tasks
- **REVOKE**: remove the user accessibility to database object

iv. **Transaction Control Language (TCL)**

It manages the transactions within a database. Transactions group a set of related tasks into a single, executable task and all the tasks must succeed in order for the transaction to work. Examples

- **commit**: saves the work done
- **savepoint**: identifies a point in a transaction to which you can later roll back
- **rollback**: restores the database to original since the last COMMIT

This lab manual is designed to help students to understand basic commands of DDL, DML, DCL using command line as well as using any one of the Integrated Development Environment such as MYSQL Workbench, Phpmyadmin. Students will learn to import existing databases and export their databases for schema transfer or backup purpose.This lab manual tries to cover the most of the theoretical portion that students have learnt during the course of CT652 Database Management System.

Lab Work: 1

Title: Understanding DDL operations from Command Line

Objective:

- To understand basic DDL commands through command line

Theoretical Background:

Command Line Login to MYSQL

Find the path of bin directory of MYSQL server installed in your system. Open command prompt and Login to the MYSQL server using the following command:

```
:\.....\mysql\bin> mysql –u username –p;
```

Type the user's password, and then press Enter

For Example: Login with root user : ..\bin>mysql –uroot –p;

To create a database, type the following command. Replace *dbname* with the name of the database that you want to create:

```
Mysql>create database dbname;
```

To view the existing databases in the database server, use the following command

```
Mysql> show databases;
```

From the list of databases, select a database that you want to work with, using the following command

```
Mysql> use dbname;
```

To view existing tables in the current database

```
>show tables;
```

Syntax to create a table:

create table *table_name* (A_1 D_1, A_2 D_2, ..., A_n D_n,
(integrity-constraint$_1$),
...,
(integrity-constraint$_k$))

- Where each A_i is an attribute name in the schema of relation *table_name and* D_i is the data type of values in the domain of attribute A_i

Example:

```
CREATE TABLE student (sid INT(2), sname VARCHAR(20), PRIMARY KEY (sid));
```

Or

CREATE TABLE student (sid INT(2) primary key, sname VARCHAR(20), PRIMARY KEY (sid));

- ➔ DESC student; It describes schema of the table student
- ➔ Add column to the table
    - o ALTER TABLE student
      ADD email VARCHAR(40);
- ➔ Delete Column
    - o ALTER TABLE student
      DROP COLUMN email;
- ➔ Edit datatype of some columns
    - o ALTER TABLE student
      MODIFY COLUMN sname VARCHAR(40);
- ➔ Alter column name is not advisable since there might be dependencies on that columns or field name
  Syntax: ALTER TABLE table_name RENAME COLUMN old_col_name TO new_col_name;
  Example: ALTER TABLE student RENAME COLUMN sname to stuname;

- ➔ Alter table name
    - o ALTER TABLE student
      RENAME TO tbl_student;

In-Lab Activities:

1. Login to your MySQL server through command line
2. View existing databases in your database server
3. Create your own database DB_YourCRN
4. Use your database and Create the following table with the given schema
   Employee (eid, ename, eaddress)
5. Add a new attribute ‘department_name’ to the Employee table
6. Rename column name ‘eaddress’ to ‘eprovince’ and change its datatype from varchar to integer
7. Show existing tables in your database
8. View schema of the table ‘Employee’

Lab Work: 2

Title: Understanding SQL constraints as a part of DDL and basic DML queries

Objectives:

- To learn to apply different types of SQL constraints
- To understand basic DML queries

In-Lab Activities

Consider the following relational schema

Account (<u>account_number</u>, branch_name, balance)
Branch (<u>branch_name</u>, branch_city, assets)
Customer (<u>customer_name</u>, customer_street, customer_city)
Loan (<u>loan_number</u>, branch_name, amount)
Depositor (<u>customer_name</u>, <u>account_number</u>)
Borrower (<u>customer_name</u>, <u>loan_number</u>)

1. Create the tables as mentioned in the above schema. Show the use of primary key and foreign key, unique key and not null constraints
2. Insert at least five records in each relations
3. Write an SQL query to list the names of all depositors along with their account number and balance
4. Write an SQL query to find the names of all customers who have a loan of over 150,000
5. Write a query in SQL to increase all accounts with balances over 100,000 by 6% and all other accounts by 5%
6. Show all the constraints used in the table 'Account'
7. Delete foreign key constraint that is used on 'account_number' field of the table 'Depositor'
8. Add a foreign key constraint 'fk_depositor_1' on field 'account_number' of the table 'Depositor'.

Lab Work: 3

Title: Database Import and DML Queries

Objectives:

- To import database from the existing sql dump file
- To understand the imported database through DML queries

Importing Database *Classic Models*

ClassicModels is a MYSQL sample database which can be downloaded freely from the internet or your lab instructor will provide you .sql dump database

Procedure to Import database from .sql file

Login to your mysql server in command prompt

```
C:\........../bin> mysql –u root –p;
```

```
mydql>source sampledatabase.sql;
```

If the .sql file is in other location than the mysql/bin directory then you have to give full path of the file as shown below:

```
mysql> source F:/db/sampledatabase.sql;
```

About MySQL Sample Database: Classic Model

The Classic Models database is a retailer of scale models of classic cars. The sample database contains typicalbusiness data such as customers, products, sale orders, sale order line items and etc.

Sample Database Schema

The sample database schema consists of several table as below:

Customers: Stores customers's data

Products: Stores a list of scale model cars.

ProductLines: Stores a list of product line category.

Orders: Stores orders placed by customers.

OrderDetails: Stores order line items in each order.

Payments: Stores payments made by customers based on their account.

Employees: Stores all employee information include organization unit structure such as who reports to whom.

Offices: Stores sale office data.

Using MySQL SELECT Statement to Query Data

```
SELECT column_name1,column_name2...
FROM tables
[WHERE conditions]
[GROUP BY group
[HAVING group_conditions]]
[ORDER BY sort_columns]
[LIMIT limits];
```

The order of FROM, WHERE, GROUP BY, HAVING, ORDER BY and LIMIT has to be in the sequence above. To select all columns in a table you can use asterisk (*) notation instead of listing all column names in the MySQL SELECT statement.

In-Lab Activities:

Import the classic model database in your Mysql server. Try to understand the following queries, run them and write down what each query does.

```
1. SELECT * FROM employees
2. SELECT lastname,firstname,jobtitle
       FROM employees
3. SELECT firstname,lastname,email
       FROM employees
       WHERE jobtitle="president"
4. SELECT DISTINCT jobTitle FROM employees;
5. SELECT firstname,lastname, jobtitle
       FROM employees
       ORDER BY firstname ASC,jobtitle DESC;
6. SELECT DISTINCT city, state
       FROM customers
7. SELECT firstname,lastname
       FROM employees
       LIMIT 5
```

```
8. SELECT firstname,lastname
        FROM employees
        LIMIT 10,5
9. SELECT officeCode, city, phone
        FROM offices
        WHERE country IN ('USA','France')
10. SELECT officeCode, city, phone
        FROM offices
        WHERE country IN ('USA','France')
11. SELECT orderNumber
        FROM orderDetails
        GROUP BY orderNumber
        HAVING SUM (quantityOrdered * priceEach) > 60000
12. SELECT orderNumber,customerNumber,status,shippedDate
        FROM orders
        WHERE orderNumber IN (
        SELECT orderNumber
        FROM orderDetails
        GROUP BY orderNumber
        HAVING SUM(quantityOrdered * priceEach) > 60000)
13. SELECT employeeNumber, lastName, firstName
        FROM employees
        WHERE firstName LIKE 'a%'
14. SELECT employeeNumber, lastName, firstName
        FROM employees
        WHERE lastName LIKE '%on'
15. SELECT employeeNumber, lastName, firstName
        FROM employees
        WHERE lastName NOT LIKE 'B%'
```

16. SELECT customerNumber id, contactLastname name

```
FROM customers
UNION
SELECT employeeNumber id,firstname name
FROM employees
```

17. (SELECT customerNumber id,contactLastname name

```
FROM customers)
UNION
(SELECT employeeNumber id,firstname name
FROM employees)
ORDER BY name,id
```

18. (SELECT customerNumber, contactLastname

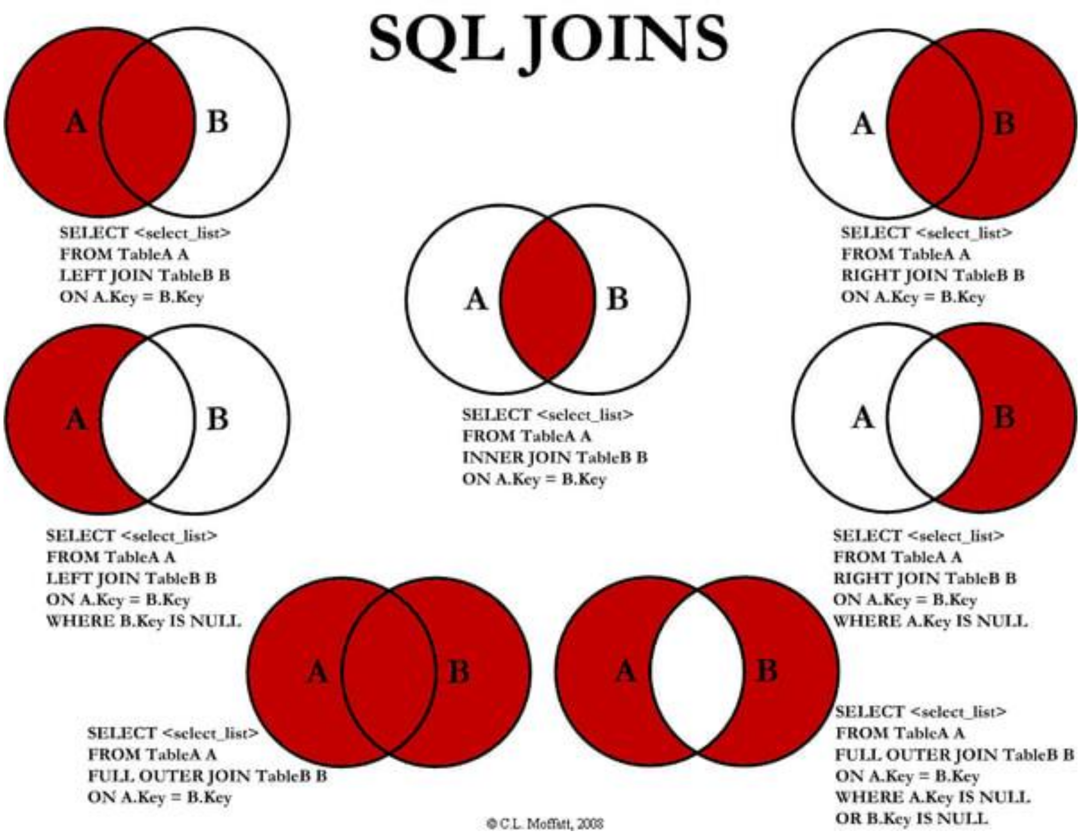
```
FROM customers)
UNION
(SELECT employeeNumber, firstname
FROM employees)
ORDER BY contactLastname, customerNumber
```

Lab Work: 4

Title: Aggregate Functions and SQL Joins

Objectives:

- To understand aggregate functions
- To be able to distinguish functionality of different types of SQL Joins



In-Lab Activities

Run the following queries over on the sample classicmodels database, understand them andexplain what each query does.

1. SELECT A.productCode, A.productName, B.orderNumberFROM products A
   INNER JOIN orderDetails B on A.productCode = B.productCode;

2. SELECT c.customerNumber, customerName,orderNumber, o.statusFROM customers c
   LEFT JOIN orders o ON c.customerNumber = o.customerNumber;

3.
   ```
   SELECT o.customerNumber, orderNumber, o.status,
   customerName FROM orders o
   RIGHT JOIN customers c ON o.customerNumber = c.customerNumber;
   ```

4.
   ```
   SELECT c.customerNumber, c.customerName, c.salesRepEmployeeNumber,
   e.lastName, e.firstName
   FROM customers c
   LEFT OUTER JOIN employees e
   ON c.salesRepEmployeeNumber = e.employeeNumber;
   ```

5.
   ```
   SELECT c.customerNumber, c.customerName, e.employeeNumber, e.lastName,
   e.firstName FROM customers c
   RIGHT OUTER JOIN employees e
   ON c.salesRepEmployeeNumber = e.employeeNumber;
   ```
6. Unfortunately, MySQL does not have a FULL OUTER JOIN. Anyway, write a query to perform customers FULL OUTER JOIN employees using an alternate way.

7.
   ```
   SELECT o.customerNumber, orderNumber, o.status,
   customerName FROM orders o
   JOIN customers c
   ```

8.
   ```
   SELECT o.customerNumber, orderNumber, o.status,
   customerName FROM orders o
   NATURAL JOIN customers c
   ```

9.
   ```
   SELECT o.customerNumber, orderNumber, o.status,
   customerName FROM orders o
   INNER JOIN customers c
   ```

10.
    ```
    SELECT customerNumber, orderNumber, status,
    customerName FROM orders
    JOIN customers
    USING (customerNumber)
    ```
11. Write a query to join the three tables - customers LEFT OUTER JOIN (orders INNER JOIN orderdetails)

12.
    ```
    SELECT customerNumber, checkNumber,
    amount FROM payments
    WHERE (customerNumber, checkNumber)
    NOT IN(
            SELECT p.customerNumber,
            p.checkNumber FROM payments p,
            payments q
            where p.amount<q.amount )
    ```

13. SELECT customerNumber,
    customerNameFROM customers
    WHERE customerName LIKE '%toys%'

14. SELECT customerNumber,
    customerNameFROM customers
    WHERE customerName COLLATE latin1_general_cs LIKE '%Land%'

15. SELECT firstName, lastName,
    extensionFROM employees
    WHERE extension LIKE 'x_ _ _ '

16. SELECT firstName, upper(lastName) as lastName FROM employees

17. SELECT p.productCode, productName FROM
    products pWHERE quantityInStock <100 AND
    EXISTS
    (SELECT orderNumber FROM orderdetails o WHERE p.productCode = o.productCode)

18. SELECT ordernumber, sum(quantityOrdered) AS itemsCount,
    sum(priceeach) AS totalFROM orderdetails
    GROUP BY ordernumber
    HAVING total > 1000 AND itemsCount > 600

19. SELECT ordernumber,
    itemsCountFROM
    (

    ```
            SELECT ordernumber, sum(quantityOrdered) AS
            itemsCountFROM orderdetails
            GROUP BY ordernumber
    ```

) as t
    WHERE t.itemsCount >300

20. Write a query to list the names of all products along with total quantity ordered for whichthe total quantity ordered has exceeded 5000

Lab Work: 5

Discussion on DBMS Project Proposal

This lab hour shall be allocated for discussion on DBMS Project Proposal. Students should submit project proposal in group. Each group shall contain three or four students. The contents of the proposal must include the following sections:

- Introduction
- Objectives
- ER diagram

User interface part is optional

Students are free to choose Case Study on existing DBMS application.

Lab Work: 6

Title: Uses of Views and Triggers

Objectives:

- To be able to create view, modify and drop views
- To understand the importance of triggers

In-Lab Activities:

Understand and run the following SQL queries. Writedown the purpose of each query.

1.

```
CREATE VIEW SalePerOrder_(your roll
 number) ASSELECT orderNumber,
 SUM (quantityOrdered * priceEach)
 totalFROM orderDetails
 GROUP by orderNumber
 ORDER BY total DESC
```

2.

```
SELECT total
FROM salePerOrder_(your roll number)
WHERE orderNumber  10102
```

3.

```
CREATE VIEW vwProducts_(your
roll numer) ASSELECT
productCode, productName,
buyPrice FROM products
WHERE buyPrice > (
      SELECT AVG  (buyPrice) FROM  products
      )
ORDER BY buyPrice DESC
```

4.

```
CREATE VIEW officeInfo_(your roll numer)AS
SELECT officeCode, phone, city
FROM offices
```

5.

```
UPDATE officeInfo
SET phone = 'your phone number'
```

```
            WHERE officeCode = 4
```

6.

```
    CREATE VIEW organization_(your roll number) AS
       SELECT CONCAT (E.lastname,E.firstname) AS Employee,
          CONCAT  (M.lastname,M.firstname) AS Manager
       FROM employees AS E INNER JOIN employees AS M
         ON M.employeeNumber = E.ReportsTo
         ORDER BY Manager
```

7.

```
            SHOW CREATE VIEW organization_(your roll number)
```

8.

```
        ALTER VIEW organization_(your roll number)
         AS
         SELECT CONCAT(E.lastname,E.firstname) AS Employee,
              E.email AS employeeEmail,
              CONCAT(M.lastname,M.firstname) AS Manager
         FROM employees AS E INNER JOIN employees
         AS M
          ON M.employeeNumber = E.ReportsToORDER BY Manager
```

9.

```
            DROP VIEW IF EXISTS organization_(your roll number)
```

10.

```
            CREATE TABLE employees_audit (
            id int(11) NOT NULL AUTO_INCREMENT,
            employeeNumber int(11) NOT NULL, lastname
            varchar(50) NOT NULL, changedon datetime
            DEFAULT NULL, action varchar(50) DEFAULT
            NULL, PRIMARY KEY (id)
            )
```

11. Set the delimiter to | before running the following query

```
CREATE TRIGGER before_employee_update_(your roll number)
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
INSERT INTO employees_audit
SET action = 'update',
employeeNumber = OLD.employeeNumber,lastname =
OLD.lastname,
changedon = NOW();
END|
```

Update a record in the employees table and check the employees_audit table.Why do you think we need to change the delimiter?

12.

```
SELECT *
FROM Information_Schema.Triggers
WHERE Trigger_schema = 'classicmodels'
AND Trigger_name = 'before_employee_update_(your roll number)';
```

13.

```
DROP TRIGGER before_employee_update_(your roll number)
```

Lab Work: 7

Title: Understanding basic implementation of Procedural Structured Query Language (PL/SQL)

Objectives:

- To be able to create function in PL/SQL
- To be able to create procedure
- To understand difference between function and procedure

Theoretical Background:

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements. All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

```
DECLARE
  Declaration statements;
BEGIN
  Execution statements;
 EXCEPTION
    Exception handling statements;
END;
```

Procedures

A SQL is similar to a procedure in other programming languages which performs one or more tasks. A procedure has a header and a 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.

Parameters	**Description**
IN type	These types of parameters are used to send values to stored procedures.
OUT type	These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.
IN OUT type	These types of parameters are used to send values and get values from stored procedures.

A procedure may or may not return any value.

Syntax:

```
CREATE [OR REPLACE] PROCEDURE procedure_name (<Argument> {IN, OUT, IN
OUT}    <Datatype>,…)
IS
  Declaration section<variable, constant> ;
BEGIN
  Execution section
EXCEPTION
  Exception section
END
```

Two ways to execute a procedure :

- From the SQL prompt : EXECUTE [or EXEC] procedure_name;

Within another procedure – simply use the procedure name : procedure_name;

Functions

A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

Syntax:

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
 RETURN return_datatype;  {IS, AS}
 Declaration_section <variable,constant> ;
 BEGIN
    Execution_section
    Return return_variable;
 EXCEPTION
   exception section
    Return return_variable;
 END;
```

RETURN TYPE: The header section defines the return type of the function. The return datatype can be any of the oracle datatype like varchar, number etc.

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

A function can be executed in the following ways.

- As a part of a SELECT statement : SELECT emp_details_func FROM dual;
- In a PL/SQL Statements like, : dbms_output.put_line(emp_details_func);

This line displays the value returned by the function .

In-Lab Activities:

1. Create a table **Student** havinf two fields sid and batch with number datatype.

```
CREATE OR REPLACE PROCEDURE p_student(sid IN NUMBER, batch IN NUMBER)
AS
BEGIN
  INSERT INTO student VALUES(sid, batch);
  DBMS_OUTPUT.PUT_LINE('One record inserted into Student.');
END;
/
```

To run the procedure, follow the following steps

- save the procedure at d://p_stu.sql
- SQL>set serveroutput on
- SQL> start d://p_stu.sql   // procedure will be created
- SQL> exec p_student(10, 2075); //One record inserted into Student

2. Create a function as shown below and understand it

```
create or replace function fn_getbatch (id IN number) return number
is  bch number(5);
begin
                select batch into bch from student where sid=id;
 return bch;
end;
/
```

To run the function, follow the following steps

- Save the function at d://f_stu.sql
- SQL> start d://f_stu.sql ;   // Function will be created
- Select fn_getbatch(10) from DUAL;

  This will display 2075 as we have inserted the data in problem 1.

3. Destroy the created procedure and function using the syntax below

Syntax:

```
DROP PROCEDURE/FUNCTION PROCEDURE/FUNCTION_NAME;
```

Lab Work: 8

Title: Understanding Data Control Language (DCL)

Objectives:

- To understand grant and revoke privileges on database objects

Theoretical Background:

Data Control Language (DCL) consists of various commands which are related to data sharing and security of data in database.They are

GRANT

REVOKE

Granting Privileges:

Objects that are created by a user are owned and controlled by that user. If user wishes to access any of the objects belonging to another user, the owner of the object will have to give permissions for such access. This is called Granting of Privileges.

Granting privileges using the GRANT statements:

The GRANT statements provide various types of access to database objects such as tables, views.

Syntax:

```
GRANT {object privileges}
ON object name
TO username;
```

The list of object privileges is as follows:

• **ALTER:** allows the grantee to change the table definitions with the ALTER table command.

• **DELETE:** allows the grantee to remove the records from the table with the DELETE command.

• **INDEX:** allows the grantee to create an index on the table with the CREATE INDEX command.

• **INSERT:** allows the grantee to add records to the table with the INSERT command.

• **SELECT:** allows the grantee to query the table with SELECT command.

• **UPDATE:** allows the grantee to modify the records in the table with the UPDATE command.

Revoking privileges:

Privileges once given can be denied to a user using the REVOKE command. The object owner can revoke privileges granted to another user. A user of an object who is not owner, but has been granted the GRANT privilege, has the power to REVOKE the privileges from the grantee.

Revoking permission using the REVOKE statement:

The REVOKE statement is used to deny the grant given on an object.

Syntax:

```
REVOKE {object privileges}
ON object name
FROM username;
```

The REVOKE command cannot be used to revoke the privileges granted through operating system

In-Lab Activities:

1. Create your own database
2. Create database users
3. Assign users to particular database
4. Grant privilege to access certain table of one schema to another schema (supports modular programming)
5. Revoke privileges

DBMS Project Presentation

Day 1: Project Presentation, Demonstration and Viva

Day 2: Project Presentation, Demonstration and Viva