RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS23332 DATABASE MANAGEMENT SYSTEMS LAB

Laboratory Record Note Book

Name : LOKAA.V

Year / Branch / Section : AIML-B (II)

University Register No. : 211623150108S

College Roll No. : 231501085

Semester : 3

Academic Year : 2024.

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

BONAFIDE CERTIFICATE

NAME LOKAA.V

ACADEMIC YEAR 2024 SEMESTER 3 BRANCH B.Tech CA INL)

UNIVERSITY REGISTER No. 2116231501085

Certified that this is the bonafide record of work done by the above student in the DATABASE MANAGEMENT SYSTEM [CS23332] Laboratory during the year 20 - 20

Signature of Faculty - in - Charge

Submitted for the Practical Examination held on .....

Internal Examiner

External Examiner

INDEX PAGE

SL.NO	DATE	NAME OF THE EXPERIMENT	PAGE NO	MARKS	FACULTY'S SIGNATURE
01	26/7/24	CREATION OF BASE TABLE AND DML OPERATIONS	6	14	
02	30/7/24	DATA MANIPULATIONS	10	14	
03	2/7/24	WRITING BASIC SQL SELECT STATEMENT	15	14	
04	6/7/24	WORKING WITH CONSTRAINTS	19	14	
05	13/8/24	CREATING VIEWS	26	14	
06	20/8/24	RESTRICTING AND SORTING DATA	31	14	
07	27/8/24	USING SET OPERATORS	38	14	
08	30/9/24	WORKING WITH MULTIPLE TABLES	42	14	
09	10/9/4	SUB QUERIES	52	14	
10	20/9/24	AGGREGATING DATA USING GROUP FUNCTIONS	59	14	
11	27/9/24	PL SQL PROGRAMS	67	14	
12	27/9/24	WORKING WITH CURSOR PROCEDURES AND FUNCTIONS	83	14	
13	1/10/24	WORKING WITH TRIGGER	93	14	
14	8/10/24	MONGO DB	101	14	
15	18/10/24	OTHER DATABASE OBJECTS	112	14	
16	25/11/24	CONTROLLING USER ACCESS	118	14	

Ex.No.: 1	CREATION OF BASE TABLE AND DML OPERATIONS
Date: 26/7/24	

AIM:

ALGORITHM:

STEP-1: Start.

STEP-2: Create a base Table

Syntax:

CREATE TABLE <table name> (column1 type, column2 type, ...);

STEP-3: Describe the Table structure

Syntax:

DESC <table name>

STEP-4: Add a new row to a Table using INSERT statement.

Syntax:

- INSERT INTO <table name> VALUES (value1, value2..);
- INSERT INTO <table name> (column1, column2..)
  VALUES (value1, value2..);
- INSERT INTO <table name>VALUES (&column1,'&column');

STEP-5: Modify the existing rows in the base Table with UPDATE statement.

Syntax:

UPDATE <table name> SET column1=value, column2 = 'value'
WHERE (condition);

STEP-6: Remove the existing rows from the Table using DELETE statement.

Syntax:

DELETE FROM <table name> WHERE <condition>;

STEP-7: Perform a Query using SELECT statement.

Syntax:

SELECT [DISTINCT] {*,<column1,,..>} FROM <table name>
WHERE <condition>;

3.

ID	Last_name	FIRST_name	USER-ID	SALARY.
1.	Patel	Ralph	rpatel	895
2.	Danes	Betty	bdanes	860

4.

ID	FIRST_name	Last_name	USER-ID	SALARY.
1.	Patel	Ralph	rpatel	895
2.	Danes	Betty	bdanes	860
3.	Biri	Ben	bbiri	1100
4.	Newman	Chad	cnewman	750

5.

ID	Last_name	First_name	User ID	Salary.
1.	Patel	Ralph	rpatel	895
3.	Biri	Ben	bbiri	1100
4.	Newman	Chad.	cnewman	750

NAME	NULL?	TYPE
ID	Not null	Number(4)
Last_name		Varchar(25)
First_name		Varchar(25)
Userid		Varchar(25)
Salary		Number(9,2)

2. Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

insert into my-employee values (1, 'Patel', 'Ralph', 'rpatel', 895);

insert into my-employee values (2, 'Dancs', 'Betty', 'bdancs', 860);

3. Display the table with values.

select * from my-employee ;

4. Populate the next two rows of data from the sample data. Concatenate the first letter of the first_name with the first seven characters of the last_name to produce Userid.

insert into my-employee values (3, 'Biri', 'Ben', 'bbiri', 1100);

insert into my-employee values (4, 'Newman', 'chad', 'cnewman', 750);

select * from my-employee ;

5. Delete Betty dancs from MY _EMPLOYEE table.

delete from my-employee where ID = 2

6. Empty the fourth row of the emp table.

delete from my_employee where ID=4;

7. Make the data additions permanent.

commit;

8. Change the last name of employee 3 to Drexler.

update my_employee set last_name = "Drexler"
where ID = 3;

9. Change the salary to 1000 for all the employees with a salary less than 900.

update my_employee set salary = 1000 where salary<900;

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 2	
Date:	30/7/24

DATA MANIPULATIONS

Create the following tables with the given structure.

EMPLOYEES TABLE

NAME	NULL?	TYPE
Employee_id	Not null	Number(6)
First_Name		Varchar(20)
Last_Name	Not null	Varchar(25)
Email	Not null	Varchar(25)
Phone_Number		Varchar(20)
Hire_date	Not null	Date
Job_id	Not null	Varchar(10)
Salary		Number(8,2)
Commission_pct		Number(2,2)
Manager_id		Number(6)
Department_id		Number(4)

(a) Find out the employee id, names, salaries of all the employees

```
Select employee_id , first_name, last_name, salary
from employees ;
```

(b) List out the employees who works under manager 100

```
Select employee_id , first_name, last_name
from employees where manager_id =100;
```

(c) Find the names of the employees who have a salary greater than or equal to 4800

7 Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

comment on table dept is 'Table containing dept details';
comment on table emp is 'Table containing emp details';

8 Drop the First_name column from the EMP table and confirm it.

alter table emp drop (column first_name);

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 3		WRITING BASIC SQL SELECT STATEMENTS
Date:	2/7/24	

OBJECTIVES

After the completion of this exercise, the students will be able to do the following:

- List the capabilities of SQL SELECT Statement
- Execute a basic SELECT statement

Capabilities of SQL SELECT statement

A SELECT statement retrieves information from the database. Using a select statement, we can perform

- ✔ Projection: To choose the columns in a table
- ✔ Selection: To choose the rows in a table
- ✔ Joining: To bring together the data that is stored in different tables

Basic SELECT Statement

Syntax

```
SELECT *|DISTINCT Column_ name| alias
    FROM table_name;
```

NOTE:

DISTINCT—Suppress the duplicates.

Alias—gives selected columns different headings.

Example: 1

```
SELECT * FROM departments;
```

Example: 2

```
SELECT location_id, department_id FROM departments;
```

Writing SQL Statements

- SQL statements are not case sensitive
- SQL statements can be on one or more lines.

3. Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first.

```
select employee_num, last_name, job_code, hire_date
    from employees;
```

4. Provide an alias STARTDATE for the hire date.

```
select employee_num, last_name, job_code, hire_date
    as startdate from employees;
```

5. Create a query to display unique job codes from the employee table.

```
select distinct job_code
    from employees;
```

6. Display the last name concatenated with the job ID , separated by a comma and space, and name the column EMPLOYEE and TITLE.

```
select CONCAT (last_name, ', ', job_id) As
    Employee_And_Title from employees;
```

7. Create a query to display all the data from the employees table. Separate each column by a comma. Name the column THE_OUTPUT.

```
select . concat_ws (',', employee_num, last_name,
    job_code, hire_date) As the_output
        from employees;
```

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 4	WORKING WITH CONSTRAINTS
Date: 6/7/24	

OBJECTIVE

After the completion of this exercise the students should be able to do the following

- Describe the constraints
- Create and maintain the constraints

What are Integrity constraints?

- Constraints enforce rules at the table level.
- Constraints prevent the deletion of a table if there are dependencies

The following types of integrity constraints are valid

a) **Domain Integrity**

✔ NOT NULL
✔ CHECK

b) **Entity Integrity**

✔ UNIQUE
✔ PRIMARY KEY

c) **Referential Integrity**

✔ FOREIGN KEY

Constraints can be created in either of two ways

1. At the same time as the table is created
2. After the table has been created.

Defining Constraints

Create table tablename (column_name1 data_ type constraints, column_name2 data_ type constraints …);

Example:

Create table employlees (employee_id number(6), first_name varchar2(20), ..job_id varchar2 (10), CONSTRAINT emp_emp_id_pk PRIMARY KEY (employlee_id));

4. Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

Alter table emp add commision number(2,2)
check(commision > 0);

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 5	CREATING VIEWS
Date: 13/8/24	

After the completion of this exercise, students will be able to do the following:

- Describe a view
- Create, alter the definition of, and drop a view
- Retrieve data through a view
- Insert, update, and delete data through a view
- Create and use an inline view

View

A view is a logical table based on a table or another view. A view contains no data but is like a window through which data from tables can be viewed or changed.The tables on which a view is based are called base tables.

Advantages of Views

- To restrict data access
- To make complex queries easy
- To provide data independence
- To present different views of the same data

Classification of views

1. Simple view
2. Complex view

Feature	Simple	Complex
No. of tables	One	One or more
Contains functions	No	Yes
Contains groups of data	No	Yes
DML operations thr' view	Yes	Not always

Creating a view

Syntax

5. Create a view named DEPT50 that contains the employee number, employee last names and department numbers for all employees in department 50.Label the view columns EMPNO, EMPLOYEE and DEPTNO. Do not allow an employee to be reassigned to another department through the view.

create view dept 50 As select employee_id As emp no last_name As employee, department_id as DEPT NO from employees where department_ID =50 with check OPTION;

6. Display the structure and contents of the DEPT50 view.

Describe DEPT 50;
select * from DEPT50;

7. Attempt to reassign Matos to department 80.

update DEPT50. SET DEPT NO =80 where employee = "Matos";

8. Create a view called SALARY_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the Employees, DEPARTMENTS and JOB_GRADE tables. Label the column Employee, Department, salary, and Grade respectively.

create view salary_vu as select e.last_name As employee, d.department name as department, e.salary As salary, j.grade As grade.

from employees e
join department d
on e.department_id
= d.department_id
join job_grade j
on e.salary
between j.lowest_salary and
j.highest salary;

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 6	RESTRICTING AND SORTING DATA
Date: 20/8/24	

After the completion of this exercise, the students will be able to do the following:

- Limit the rows retrieved by the queries
- Sort the rows retrieved by the queries
-

Limiting the Rows selected

- Using WHERE clause
- Alias cannot used in WHERE clause

Syntax

SELECT----------
FROM----------
WHERE condition;

Example:

SELECT employee_id,last_name, job_id, deparment_id FROM employees WHERE department_id=90;

Character strings and Dates

Character strings and date values are enclosed in single quotation marks.

Character values are case sensitive and date values are format sensitive.

Example:

SELECT employee_id,last_name, job_id, deparment_id FROM employees
WHERE last_name='WHALEN";

Comparison Conditions

All relational operators can be used. (=, >, >=, <, <= ,<>,!=)

Example:

SELECT last_name, salary

9. Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.(hints: is not nul,orderby)

```
select last_name , salary, commission_pct
from employees
where commission_pct is not null
order by salary Desc, commision_pct Desc ;
```

10. Display the last name of all employees where the third letter of the name is *a*.(hints:like)

```
select last_name
from employees
where last_name like "_a%";
```

11. Display the last name of all employees who have an a and an *e* in their last name.(hints: like)

```
select last_name
from employees
where last_name like "%a%" AND last_name like "%e";
```

12. Display the last name and job and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to 2500 ,3500 or 7000.(hints:in,not in)

```
select last_name "Employee", salary "Monthly salary"
from employees
where commission_pct = .20;
```

commission_pct

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 7	USING SET OPERATORS
Date: 27/8/24	

Objectives

After the completion this exercise, the students should be able to do the following:

- Describe set operators
- Use a set operator to combine multiple queries into a single query
- Control the order of rows returned

The set operators combine the results of two or more component queries into one result.

Queries containing set operators are called *compound queries*.

Operator	Returns
UNION	All distinct rows selected by either query
UNION ALL	All rows selected by either query, including all duplicates
INTERSECT	All distinct rows selected by both queries
MINUS	All distinct rows that are selected by the first SELECT statement and not selected in the second SELECT statement

The tables used in this lesson are:

- EMPLOYEES: Provides details regarding all current employees

- JOB_HISTORY: Records the details of the start date and end date of the former job, and the job identification number and department when an employee switches jobs

UNION Operator

Guidelines

- The number of columns and the data types of the columns being selected must be identical in all the SELECT statements used in the query. The names of the columns need not be identical.

- UNION operates over all of the columns being selected.

- NULL values are not ignored during duplicate checking.

- The IN operator has a higher precedence than the UNION operator.

3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using set operators.

4. Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

5. The HR department needs a report with the following specifications:

- Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department.

- Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them Write a compound query to accomplish this.

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

10. Create a query to display the name and hire date of any employee hired after employee Davies.

11. Display the names and hire dates for all employees who were hired before their managers, along with their manager's names and hire dates. Label the columns Employee, Emp Hired, Manager, and Mgr Hired, respectively.

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 9	SUB QUERIES
Date: 10/9/24	

Objectives

After completing this lesson, you should be able to do the following:
- Define subqueries
- Describe the types of problems that subqueries can solve
- List the types of subqueries
- Write single-row and multiple-row subqueries

Using a Subquery to Solve a Problem
Who has a salary greater than Abel's?

Main query:
Which employees have salaries greater than Abel's salary?

Subquery:
What is Abel's salary?

Subquery Syntax

SELECT *select_list* FROM *table* WHERE *expr operator* (SELECT *select_list* FROM *table*);

- The subquery (inner query) executes once before the main query (outer query).

- The result of the subquery is used by the main query.

A subquery is a SELECT statement that is embedded in a clause of another SELECT statement. You can build powerful statements out of simple ones by using subqueries. They can be very useful when you need to select rows from a table with a condition that depends on the data in the table itself.

You can place the subquery in a number of SQL clauses, including the following:

- WHERE clause
- HAVING clause
- FROM clause

In the syntax:
operator includes a comparison condition such as >, =, or IN

Note: Comparison conditions fall into two classes: single-row operators

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 10	AGGREGATING DATA USING GROUP FUNCTIONS
Date: 20/9/24	

Objectives

After the completion of this exercise, the students be will be able to do the following:

- Identify the available group functions
- Describe the use of group functions
- Group data by using the GROUP BY clause
- Include or exclude grouped rows by using the HAVING clause

What Are Group Functions?

Group functions operate on sets of rows to give one result per group

Types of Group Functions

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Each of the functions accepts an argument. The following table identifies the options that you can use in the syntax:

Function	Description
AVG([DISTINCT|ALL] *n*)	Average value of *n*, ignoring null values
COUNT({*| [DISTINCT|ALL] *expr*})	Number of rows, where *expr* evaluates to something other than null (count all selected rows using *, including duplicates and rows with nulls)
MAX([DISTINCT|ALL] *expr*)	Maximum value of *expr*, ignoring null values
MIN([DISTINCT|ALL] *expr*)	Minimum value of *expr*, ignoring null values
STDDEV([DISTINCT|ALL] *x*)	Standard deviation of *n*, ignoring null values
SUM([DISTINCT|ALL] *n*)	Sum values of *n*, ignoring null values
VARIANCE([DISTINCT|ALL] *x*)	Variance of *n*, ignoring null values

Group Functions: Syntax

SELECT [*column*,] *group_function(column)*, ...
FROM *table*
[WHERE *condition*]

11. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

12.Write a query to display each department's name, location, number of employees, and the average salary for all the employees in that department. Label the column name-Location, Number of people, and salary respectively. Round the average salary to two decimal places.

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 11		PL SQL PROGRAMS
Date:	27/9/24	

PROGRAMS

TO DISPLAY HELLO MESSAGE

```
SQL> set serveroutput on;
SQL> declare
  2  a varchar2(20);
  3  begin
  4  a:='Hello';
  5  dbms_output.put_line(a);
  6  end;
  7  /
Hello

PL/SQL procedure successfully completed.
```

TO INPUT A VALUE FROM THE USER AND DISPLAY IT

```
SQL> set serveroutput on;
SQL> declare
  2  a varchar2(20);
  3  begin
  4  a:=&a;
  5  dbms_output.put_line(a);
  6  end;
  7  /
Enter value for a: 5
old   4: a:=&a;
new   4: a:=5;
5

PL/SQL procedure successfully completed.
```

GREATEST OF TWO NUMBERS

```
SQL> set serveroutput on;

SQL> declare
  2  a number(7);
```

PROGRAM 15
Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

```
SET SERVEROUTPUT ON;

BEGIN
FOR rec IN (SELECT e.emp_id, e.name, j.end_date
            FROM emp e
            JOIN job_history j ON e.emp_id)

END LOOP;

END;
/
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 12	WORKING WITH CURSOR, PROCEDURES AND FUNCTIONS
Date: 27/9/24	

AIM:

Create PL/SQL Blocks to perform the Item Transaction Operations using CURSOR, FUNCTION and PROCEDUERE.

ALGORITHM:

STEP-1: Start.

STEP-2: Create two tables Item Master and Item Trans.

itemmaster(itemid , itemname, stockonhand)

itemtrans(itemid ,itemname ,dateofpurchase ,quantity)

STEP-3: Create a PROCEDURE with id, name and quantity as parameters which make a call to the FUNCTION by passing id, name, dop, and quantity as parameters dop is set as sysdate.

STEP-4: Using FUNCTION fetch each record from the table Item Master using CURSOR inside a Loop statement,

If Item Master's ItemId is equal to the entered ID value then exit the loop otherwise fetch the next record.

```
loop
        fetch master into masterrec
        exit when master%notfound
        if masterrec.itemid=id then
    exit;
  end if;
end loop;
```

STEP-5: If Itemmaster's itemid = id then,

Add the Itemmaster's stockonhand with the given quantity and update the ItemMaster table and insert the Item information into the ItemTrans table.

STEP-6: Else, if the inputed item is not present in the ItemMaster table then insert the

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 13	WORKING WITH TRIGGER
Date: 1/10/24	TRIGGER

DEFINITION

A trigger is a statement that is executed automatically by the system as a side effect of a modification to the database. The parts of a trigger are,

- **Trigger statement**: Specifies the DML statements and fires the trigger body. It also specifies the table to which the trigger is associated.
- **Trigger body or trigger action**: It is a PL/SQL block that is executed when the triggering statement is used.
- **Trigger restriction**: Restrictions on the trigger can be achieved

The different uses of triggers are as follows,

- *To generate data automatically*
- *To enforce complex integrity constraints*
- *To customize complex securing authorizations*
- *To maintain the replicate table*
- To audit data modifications

TYPES OF TRIGGERS

The various types of triggers are as follows,

- **Before**: It fires the trigger before executing the trigger statement.
- **After**: It fires the trigger after executing the trigger statement
- .
- **For each row**: It specifies that the trigger fires once per row
- .
- **For each statement**: This is the default trigger that is invoked. It specifies that the trigger fires once per statement.

VARIABLES USED IN TRIGGERS

- :new
- :old

Program 8

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
CREATE OR REPLACE TRIGGER validate_stock_before_order
BEFORE INSERT ON orders
   FOR EACH ROW
BEGIN
   IF: New.order_quantity > (SELECT stock_quantity
WHERE item_id = : NEW item_id. FROM item
      END IF;
   END;
/
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 14	MONGO DB
Date: 8/10/24	

MongoDB is a free and open-source cross-platform document-oriented database. Classified as a NoSQL database, MongoDB avoids the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas, making the integration of data in certain types of applications easier and faster.

Create Database using mongosh

After connecting to your database using mongosh, you can see which database you are using by typing db in your terminal.

If you have used the connection string provided from the MongoDB Atlas dashboard, you should be connected to the myFirstDatabase database.

Show all databases

To see all available databases, in your terminal type show dbs.

Notice that myFirstDatabase is not listed. This is because the database is empty. An empty database is essentially non-existant.

Change or Create a Database

You can change or create a new database by typing use then the name of the database.

Create Collection using mongosh

You can create a collection using the createCollection() database method.

Insert Documents

insertOne()

```
db.posts.insertOne({
 title: "Post Title 1",
 body: "Body of post.",
 category: "News",
 likes: 1,
 tags: ["news", "events"],
```

released:1
directors:1
writers:1,
awards:1
genres:1
_id:0
}); }

13. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".

db.movies.find({
cast:"Charles Kayser", {title:1, lang:1, released:1,
director:1, genres:1, cast:1, _id:0});

14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that released on May 9, 1893.

db.movies.find({released:ISODate("1893-05-09T00:00:00Z")},
{title:1, lang:1, released:1, director:1, writer:1,
countries:1, _id:0});

14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have a word "scene" in the title.

db.movies.find({title:scene});
{title:1, lang:1, released:1,
director:1, writers:1,
countries:1, _id:0});

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 15	OTHER DATABASE OBJECTS
Date: 18/10/24	

OTHER DATABASE OBJECTS

Objectives

After the completion of this exercise, the students will be able to do the following:
- Create, maintain, and use sequences
- Create and maintain indexes

Database Objects

Many applications require the use of unique numbers as primary key values. You can either build code into the application to handle this requirement or use a sequence to generate unique numbers.
If you want to improve the performance of some queries, you should consider creating an index. You
can also use indexes to enforce uniqueness on a column or a collection of columns.
You can provide alternative names for objects by using synonyms.

What Is a Sequence?
A sequence:
- Automatically generates unique numbers
- Is a sharable object
- Is typically used to create a primary key value
- Replaces application code
- Speeds up the efficiency of accessing sequence values when cached in memory

The CREATE SEQUENCE Statement Syntax

Define a sequence to generate sequential numbers automatically:

CREATE SEQUENCE *sequence*
[INCREMENT BY *n*]
[START WITH *n*]
[{MAXVALUE *n* | NOMAXVALUE}]
[{MINVALUE *n* | NOMINVALUE}]
[{CYCLE | NOCYCLE}]
[{CACHE *n* | NOCACHE}];
In the syntax:
sequence is the name of the sequence generator

3. Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and
Administration. Confirm your additions. Run the commands in your script.
4. Create a nonunique index on the foreign key column (DEPT_ID) in the EMP table.

5. Display the indexes and uniqueness that exist in the data dictionary for the EMP table.

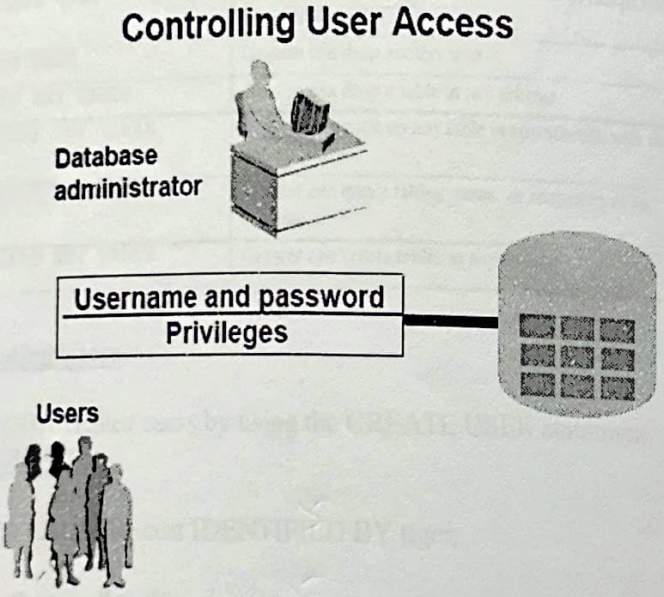
Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Ex.No.: 16	CONTROLLING USER ACCESS
Date: 25/11/24	

Objectives

After the completion of this exercise, the students will be able to do the following:

- Create users
- Create roles to ease setup and maintenance of the security model
- Use the GRANT and REVOKE statements to grant and revoke object privileges
- Create and access database links



Controlling User Access

In a multiple-user environment, you want to maintain security of the database access and use. With Oracle server database security, you can do the following:

- Control database access
- Give access to specific objects in the database
- Confirm given and received *privileges* with the Oracle data dictionary
- Create synonyms for database objects

Privileges

- Database security:
  - System security
  - Data security

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	4
Total (15)	14
Faculty Signature	

Completed