Database Management System

EXPERIMENT 14 OTHER DATABASES AND STRUCTURES

NAME : PRATHESHA J

ROLL NO : 241001172

1. Create a sequence for DEPT table primary key

```
CREATE SEQUENCE dept_id_seq
START WITH 200
INCREMENT BY 10
MAXVALUE 1000
NOCACHE
NOCYCLE;
```

Expected Output: Sequence

created.

2. Display sequence information

```
SELECT sequence_name, min_value, max_value, increment_by,
last_number
FROM user_sequences
WHERE sequence_name = 'DEPT_ID_SEQ';
```

Expected Output:

```
SEQUENCE_NAME    MIN_VALUE  MAX_VALUE INCREMENT_BY
LAST_NUMBER
```

```
-------------- ---------- ---------- ------------ ---------
--
DEPT_ID_SEQ              1       1000           10
200
```

3. Insert rows using the sequence Create the

script lab12_3.sql:

```
-- Script: lab12_3.sql
INSERT INTO dept (id, name) VALUES (dept_id_seq.NEXTVAL,
'Education');
INSERT INTO dept (id, name) VALUES (dept_id_seq.NEXTVAL,
'Administration');
COMMIT;
```

Execute the script:

```
@lab12_3.sql
```

Expected Output:

```
1 row created.

1 row created.

Commit complete.
```

Confirm the insertions:

```
SELECT * FROM dept;
```

Expected Output:

```
       ID NAME
---------- -------------------------
       200 Education        210
Administration
```

4. Create a non-unique index on foreign key column

```
CREATE INDEX emp_dept_id_idx
ON emp(dept_id);
```

Expected Output: Index

created.

5. Display indexes for EMP table

```
SELECT ic.index_name, ic.column_name, ic.column_position
col_pos, ix.uniqueness
FROM user_indexes ix, user_ind_columns ic
WHERE ic.index_name = ix.index_name
AND ic.table_name = 'EMP';
```

Expected Output:

```
INDEX_NAME      COLUMN_NAME        COL_POS UNIQUENES
--------------- ----------------- -------- ---------
EMP_DEPT_ID_IDX DEPT_ID                  1 NONUNIQUE
```

Additional Sequence and Index Examples:

Using sequences in INSERT statements

```
-- Display current sequence value
SELECT dept_id_seq.CURRVAL FROM dual;
```

Expected Output:

```
  CURRVAL
----------
       210
```

Using sequence with NEXTVAL

```
INSERT INTO dept (id, name) VALUES (dept_id_seq.NEXTVAL,
'Research');
```

Expected Output: 1 row

created.

Verify the new department:

```
SELECT * FROM dept;
```

Expected Output:

```
       ID NAME
---------- -------------------------
       200 Education
```

```
       210 Administration       220
Research
```

Creating a unique index

```
CREATE UNIQUE INDEX emp_email_idx
ON emp(email);
```

Expected Output: Index

created.

Creating a composite index

```
CREATE INDEX emp_name_idx
ON emp(last_name, first_name);
```

Expected Output: Index

created.

View all indexes in the schema

```
SELECT index_name, table_name, uniqueness
FROM user_indexes
WHERE table_name IN ('EMP', 'DEPT');
```

Expected Output:

```
INDEX_NAME      TABLE_NAME UNIQUENES
--------------- ---------- ---------
```

```
EMP_DEPT_ID_IDX EMP        NONUNIQUE
EMP_EMAIL_IDX   EMP        UNIQUE
EMP_NAME_IDX    EMP        NONUNIQUE
```

Function-based index example

```
CREATE INDEX emp_upper_name_idx
ON emp(UPPER(last_name));
```

Expected Output: Index

created.

6. Drop the sequence and indexes

```
-- Drop sequence
DROP SEQUENCE dept_id_seq;

-- Drop indexes
DROP INDEX emp_dept_id_idx;
DROP INDEX emp_email_idx;
DROP INDEX emp_name_idx;
DROP INDEX emp_upper_name_idx;
```

Expected Output:

Sequence dropped.

Index dropped.

Index dropped.

```
Index dropped.

Index dropped.
```

Performance Comparison Example:

Query without index:

```
-- Timing query without index
SET TIMING ON
SELECT * FROM emp WHERE dept_id = 20;
SET TIMING OFF
```

Create index and test again:

```
-- Create index
CREATE INDEX emp_dept_idx ON emp(dept_id);

-- Timing query with index
SET TIMING ON
SELECT * FROM emp WHERE dept_id = 20;
SET TIMING OFF
```

Key Database Objects Concepts:

Sequences:

- Automatically generate unique numbers
- Used for primary key values
- NEXTVAL: Gets next sequence value
- CURRVAL: Gets current sequence value
- CACHE: Improves performance by pre-allocating values

- CYCLE: Restarts when max value reached **Indexes:**

- Improve query performance
- Types: Unique, Non-unique, Composite, Function-based
- Automatically created for PRIMARY KEY and UNIQUE constraints
- Trade-off: Faster reads vs. slower writes **When to use indexes:**

- Columns frequently used in WHERE clauses
- Columns used in JOIN conditions
- Columns with high selectivity
- Large tables with small percentage of rows accessed **When to avoid indexes:**

- Small tables
- Columns frequently updated
- Tables with heavy write operations
- Columns with low selectivity