**USMAN INSTITUTE OF TECHNOLOGY**

**Department of Computer Science**

**CS311 Introduction to Database Systems**

Lab#10

**Objective:**

**-** **Creating sequences, indexes and synonyms**

**Name of Student: \_\_\_\_\_\_Maira Usman\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No: \_\_\_\_\_\_\_\_\_21b-011-se\_\_\_\_\_\_\_\_\_\_\_\_Sec. \_\_\_\_\_B\_\_\_\_\_\_**

**Date of Experiment: \_\_\_\_\_13-june-2023\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Marks Obtained/Remarks: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THEORY SEQUENCES**

**What is a Sequence?**

A

Sequence generator can be used to automatically generate sequence numbers for rows in tables. A sequence is a database object created by a user and can be shared by multiple users. A typical usage for sequences is to create a primary key value, which must be unique for each row. The sequence is generated and incremented (or decremented) by an internal Oracle routine. Sequence numbers are stored and generated independently of tables. Therefore, the same sequence can be used for multiple tables.

**Creating Sequences**

Following is the syntax of SQL statement to create sequences:-

CREATE SEQUENCE sequence

[INCREMENT BY n]

[START WITH n];

For example, creating a sequence named DEPT\_DEPTNO to be used for the primary key of the DEPT table.

CREATE SEQUENCE dept\_deptno

INCREMENT BY 1

START WITH 50;

**NEXTVAL and CURRVAL Pseudocolumns**

The NEXTVAL pseudocolumn is used to extract successive sequence numbers from a specified sequence. We must qualify NEXTVAL with the sequence name. When we reference *sequence*.NEXTVAL, a new sequence number is generated and the current sequence number is placed in CURRVAL.

NEXTVAL returns the next available sequence value. It returns a unique value every time it is referenced, even for different users.

CURRVAL obtains the current sequence value. NEXTVAL must be issued for that sequence before CURRVAL contains a value.

**Using a Sequence**

Insert a new department named MARKETING in San Diego

INSERT INTO dept (deptno, dname, loc)

VALUES (dept\_deptno.NEXTVAL, ‘MARKETING’, ‘SAN DIEGO’);

In order to view the current value for the DEPT\_DEPTNO sequence

SELECT dept\_deptno.CURRVAL

FORM dual;

**Removing a sequence**

A sequence can be removed by using the DROP SEQUENCE statement. Once removed, the sequence can no longer be referenced. DROP SEQUENCE dept\_deptno;

**INDEXES**

An Oracle Server index is a schema object that can speed up the retrieval of rows by using a pointer. Indexes can be created explicitly or automatically.

An index provides direct and fast access to rows in a table. Its purpose is to reduce the necessity of disk I/O by using an indexed path to locate data quickly. The index is used and maintained automatically by the Oracle Server. Once an index is created, no direct activity is required by the user.

Indexes are logically and physically independent of the table they index. Therefore, they can be created or dropped at any time and have no effect on the base tables or other indexes.

**Types of indexes**

Oracle maintains the indexes automatically: when new rows are added to the table, updated, or deleted, Oracle updates the corresponding indexes. We can create the following indexes:-

**Bitmap index**

A bitmap index does not repeatedly store the index column values. Each value is treated as a key, and for the corresponding ROWIDs a bit is set. Bitmap indexes are suitable for columns with low cardinality, such as the GENDER column in the EMP table, where the possible values are M or F. The cardinality is the number of distinct column values in a column. In the EMP table column, the cardinality of the GENDER column is 2.

**B-tree index**

This is the default. The index is created using the b-tree algorithm. The b-tree includes nodes with the index column values and the ROWID of the row. The ROWIDs are used to identify the rows in the table.

The following are the types of b-tree indexes:-

* Unique Index: The Oracle server automatically creates this index when a column in a table is defined to be a PRIMARY KEY or UNIQUE key contraint.
* NonUnique Index: Users can create nonunique indexes on columns to speed up access time to the rows. For example, we can create a FOREIGN KEY column index for a join in a query to improve retrieval speed.
* Function-based index:The function-based index can be created on columns with expressions. For example, creating an index on the SUBSTR(EMPID, 1, 2) can speed up the queries using the SUBSTR(EMPID, 1, 2) in the WHERE clause.

**Creating an Index**

* To create an index (b-tree) on ENAME column in the EMP table.

CREATE INDEX emp\_ename\_idx

ON emp(ename);

* To create an index (b-tree) on first 5 characters of JOB column in the EMP table.

CREATE INDEX emp\_job5\_idx

ON emp(SUBSTR(JOB, 1, 5));

* To create a bitmap index, we must specify the keyword BITMAP immediately after CREATE. Bitmap indexes cannot be unique. For example:

CREATE BITMAP INDEX IND\_PROJ\_STAT

ON PROJECT (STATUS);

**Confirming Indexes**

We can confirm the existence of indexes from the USER\_INDEXES data dictionary view. It contains the name of the index and its uniqueness.

SELECT INDEX\_NAME, TABLE\_NAME, TABLE\_OWNER, UNIQUENESS

FROM USER\_INDEXES;

**Removing an Index**

It is not possible to modify an index. To change it, we must drop it first and then re-create it. Remove an index definition from the data dictionary by issuing the DROP INDEX statement. To drop an index, one must be the owner of the index or have the DROP ANY INDEX privilege.

DROP INDEX *index*;

For example, remove the EMP\_ENAME\_IDX index from the data dictionary.

DROP INDEX emp\_ename\_idx;

**When to create an index**

The index should be created under following circumstances:-

* The column is used frequenctly in the WHERE clause or in a join condition.
* The column contains a wide range of values.
* The column contains a large number of null values
* Two or more columns are frequently used together in a WHERE clause or a join condition.
* The table is large and most queries are expected to retrieve less than 2-4% of the rows.

**When not to create an index**

The index should not be created under following circumstances:-

* The table is small
* The columns are not often used as a condition in the query.
* Most queries are expected to retrieve more than 2-4% of the rows.
* The table is updated frequently.

**SYNONYMS**

In order to refer to a table owned by another user, it is necessary to prefix the table name with the name of the user who created it followed by a period. Creating a synonym eliminates the need to qualify the object name with the schema and provides with an alternative name for a table, view, sequence, procedure, or other object. This method can be especially useful with lengthy object names, such as views.

The syntax is

CREATE [PUBLIC] SYNONYM *synonym*

FOR *object*;

To create a shortened name for the DEPT\_SUM\_VU view,

CREATE SYNONYM d\_sum

FOR dept\_sum\_vu;

The DBA can create a public synonym accessible to all users. e.g. to create a public synonym named DEPT for SCOTT’s DEPT table:

CREATE PUBLIC SYNONYM DEPT

FOR SCOTT.DEPT;

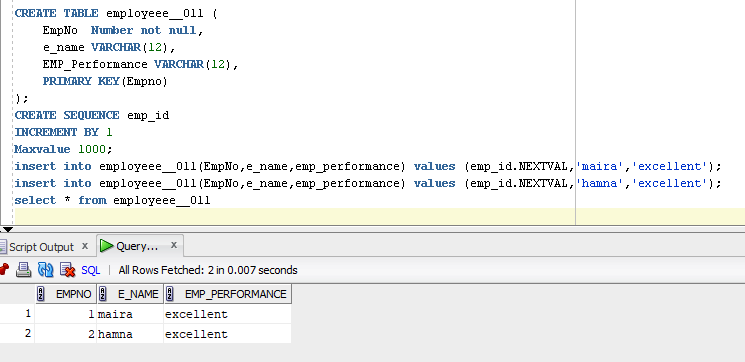
To drop a synonym,

DROP SYNONYM DEPT;

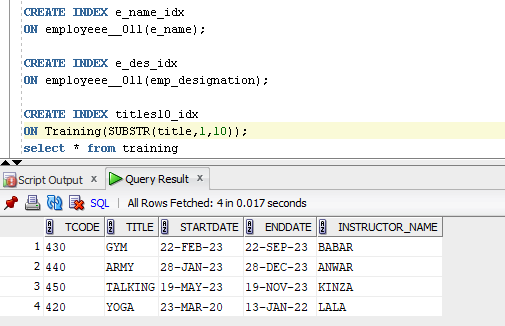
**EXERCISES**

Consider the schema of the previous lab session that represents information about *employees*, *grades*, *training* and *projects* in an organization and answer the following questions.

1. Create a sequence to generate the primary key column EMPNO of EMPLOYEE table in the lab session 06. The sequence should start with 1, increment by 1 and have maximum value of 10000.



1. Create **B-Tree** indexes on i) **Name** column of EMP table ii) **Designation** column of EMP table iii) First 10 characters of **Title** in TRAINING table



1. Create **bitmapped** indexes on i) **Gender** column of EMP table ii) **Performance** column of EMP\_PROJECT table

