USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science

CS311 Introduction to Database Systems

Lab#10

Objective:

- Creating sequences, indexes and synonyms

Name of Student: **Muhammad Waleed**

Roll No: **20B-115-SE** **Sec.** **B**

Date of Experiment:

....

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.

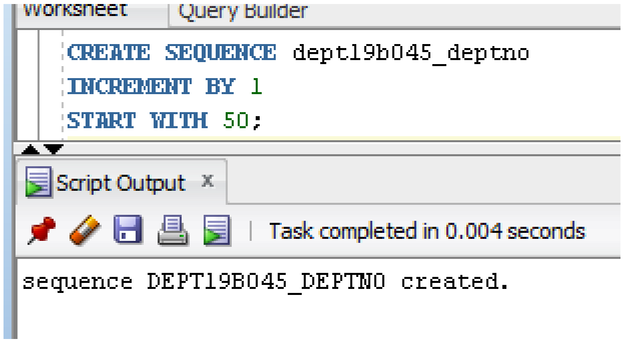
Creating Sequences

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;
```



```
CREATE SEQUENCE dept19b045_deptno
INCREMENT BY 1
START WITH 50;
```

Task completed in 0.004 seconds

```
sequence DEPT19B045_DEPTNO created.
```

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.

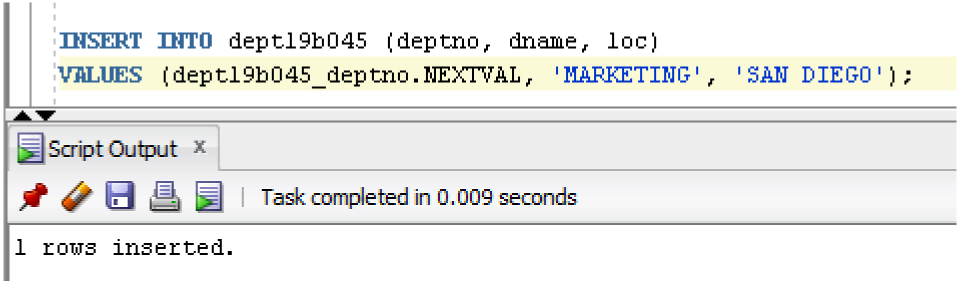
NEXTVAL returns the next available sequence value. It returns a unique value every time it is referenced, even for different users.

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’);
```

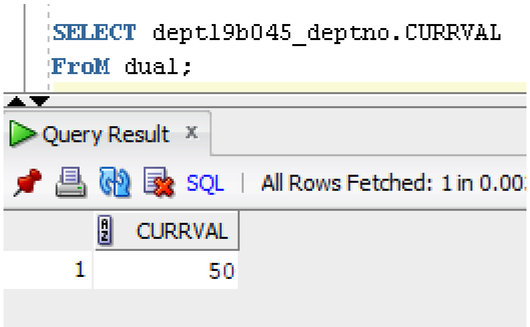


```
INSERT INTO dept19b045 (deptno, dname, loc)
VALUES (dept19b045_deptno.NEXTVAL, 'MARKETING', 'SAN DIEGO');
```

```
1 rows inserted.
```

In order to view the current value for the DEPT_DEPTNO sequence

```
SELECT dept_deptno.CURRVAL
FORM dual;
```



```
SELECT dept19b045_deptno.CURRVAL
FroM dual;
```

	CURRVAL
1	50

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

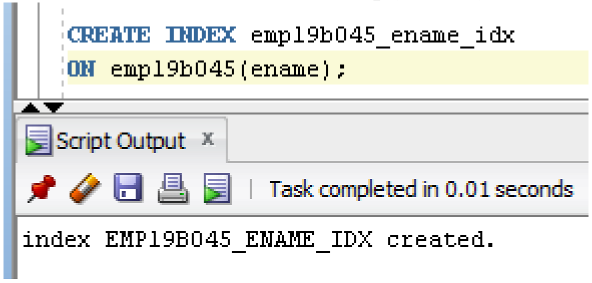
This is the default. The index is created using the b-tree algorithm. The b-tree includes nodes with the index column values and the ROWID of the row. The ROWIDs are used to identify the rows in the table. The following are the types of b-tree indexes:-

- 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);
```

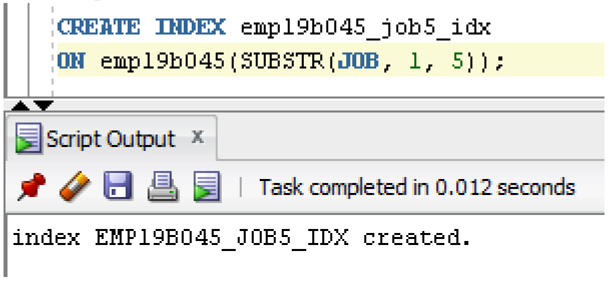


```
CREATE INDEX emp19b045_ename_idx
ON emp19b045(ename);
```

```
index EMP19B045_ENAME_IDX created.
```

- 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));
```



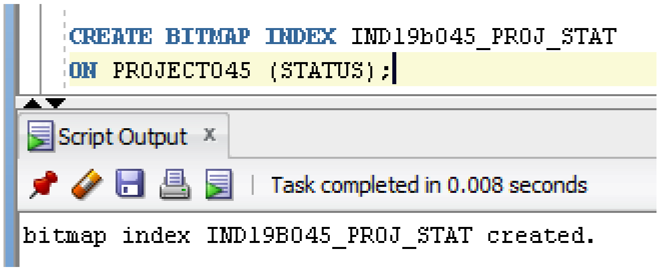
```
CREATE INDEX emp19b045_job5_idx
ON emp19b045(SUBSTR(JOB, 1, 5));
```

Task completed in 0.012 seconds

```
index EMP19B045_JOB5_IDX created.
```

- 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);
```



```
CREATE BITMAP INDEX IND19b045_PROJ_STAT
ON PROJECT045 (STATUS);
```

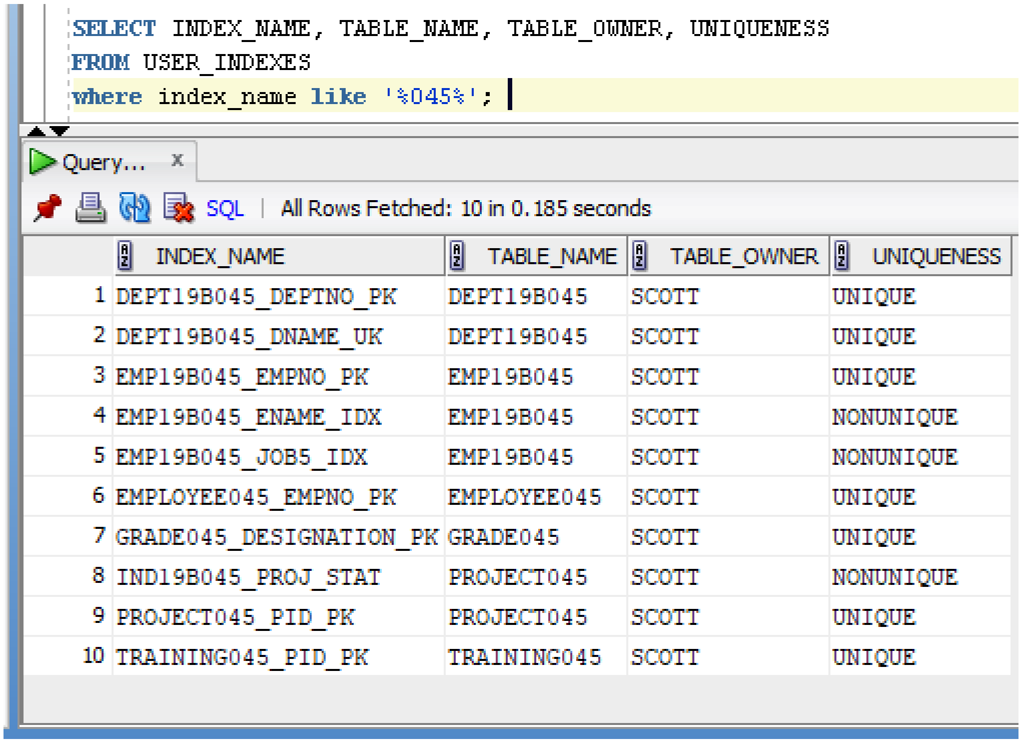
Task completed in 0.008 seconds

```
bitmap index IND19B045_PROJ_STAT created.
```

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;
```



```
SELECT INDEX_NAME, TABLE_NAME, TABLE_OWNER, UNIQUENESS
FROM USER_INDEXES
where index_name like '%045%';
```

	INDEX_NAME	TABLE_NAME	TABLE_OWNER	UNIQUENESS
1	DEPT19B045_DEPTNO_PK	DEPT19B045	SCOTT	UNIQUE
2	DEPT19B045_DNAME_UK	DEPT19B045	SCOTT	UNIQUE
3	EMP19B045_EMPNO_PK	EMP19B045	SCOTT	UNIQUE
4	EMP19B045_ENAME_IDX	EMP19B045	SCOTT	NONUNIQUE
5	EMP19B045_JOBS_IDX	EMP19B045	SCOTT	NONUNIQUE
6	EMPLOYEE045_EMPNO_PK	EMPLOYEE045	SCOTT	UNIQUE
7	GRADE045_DESIGNATION_PK	GRADE045	SCOTT	UNIQUE
8	IND19B045_PROJ_STAT	PROJECT045	SCOTT	NONUNIQUE
9	PROJECT045_PID_PK	PROJECT045	SCOTT	UNIQUE
10	TRAINING045_PID_PK	TRAINING045	SCOTT	UNIQUE

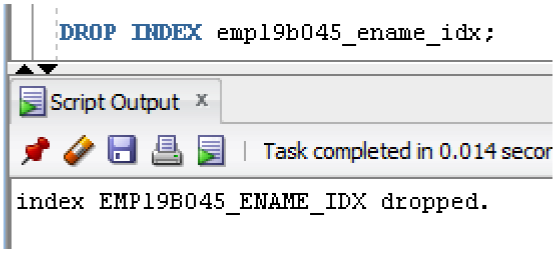
Removing an Index

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;



```
DROP INDEX emp19b045_ename_idx;
```

```
index EMP19B045_ENAME_IDX dropped.
```

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.

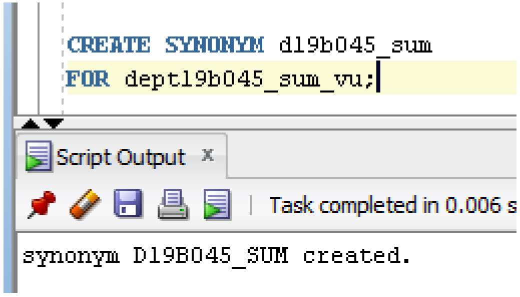
SYNONYMS

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;

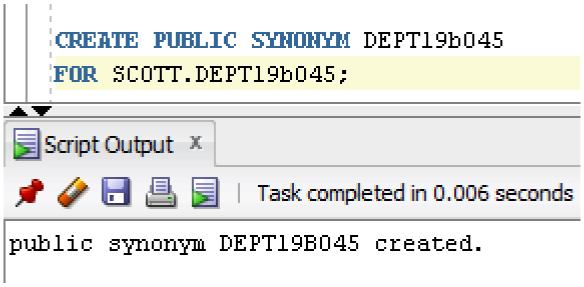


```
CREATE SYNONYM d19b045_sum
FOR dept19b045_sum_vu;
```

```
synonym D19B045_SUM created.
```

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;

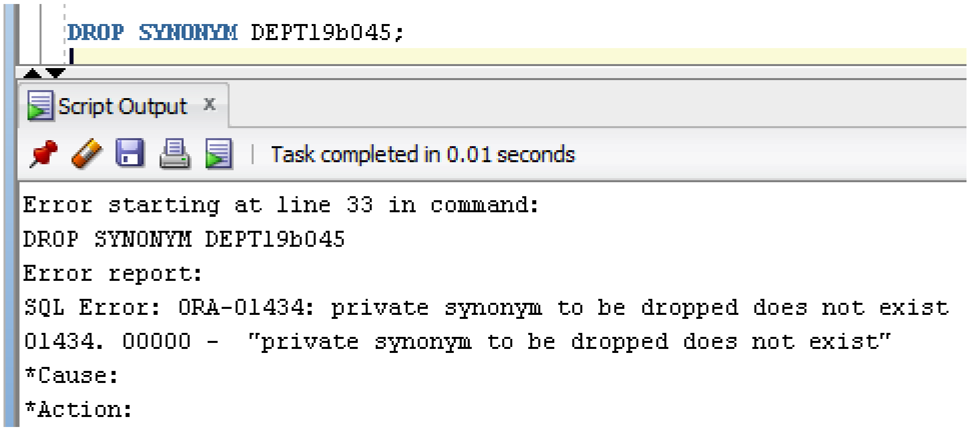


```
CREATE PUBLIC SYNONYM DEPT19b045
FOR SCOTT.DEPT19b045;
```

```
public synonym DEPT19B045 created.
```

To drop a synonym,

DROP SYNONYM DEPT;



```
DROP SYNONYM DEPT19b045;
```

```
Task completed in 0.01 seconds
```

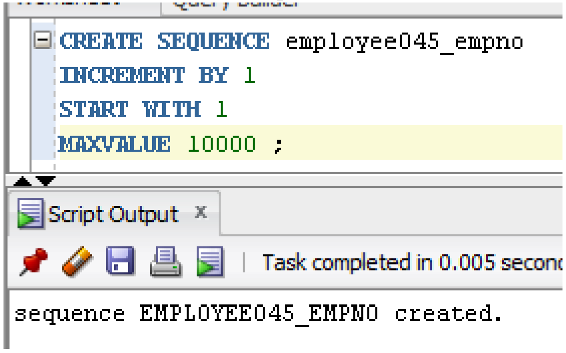
```
Error starting at line 33 in command:
DROP SYNONYM DEPT19b045
Error report:
SQL Error: ORA-01434: private synonym to be dropped does not exist
01434. 00000 -  "private synonym to be dropped does not exist"
*Cause:
*Action:
```

BECAUSE IT IS PUBLIC SYNONYMS IT CAN'T BE DROPPED

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.



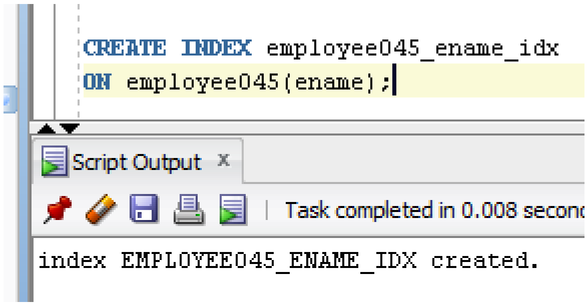
```
CREATE SEQUENCE employee045_empno
INCREMENT BY 1
START WITH 1
MAXVALUE 10000 ;
```

```
Task completed in 0.005 secon
```

```
sequence EMPLOYEE045_EMPNO created.
```

2. Create **B-Tree** indexes on

i) **Name** column of EMP table

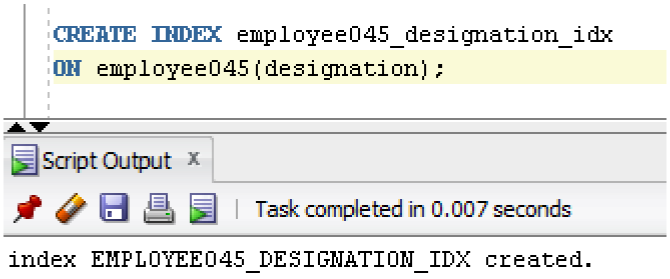


```
CREATE INDEX employee045_ename_idx
ON employee045(ename);
```

```
Task completed in 0.008 secon
```

```
index EMPLOYEE045_ENAME_IDX created.
```

ii) **Designation** column of EMP table

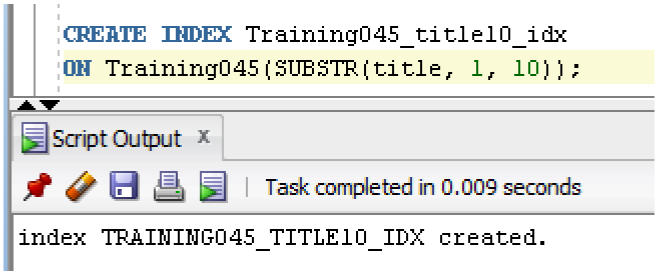


```sql
CREATE INDEX employee045_designation_idx
ON employee045(designation);
```

Task completed in 0.007 seconds

```
index EMPLOYEE045_DESIGNATION_IDX created.
```

iii) First 10 characters of **Title** in TRAINING table



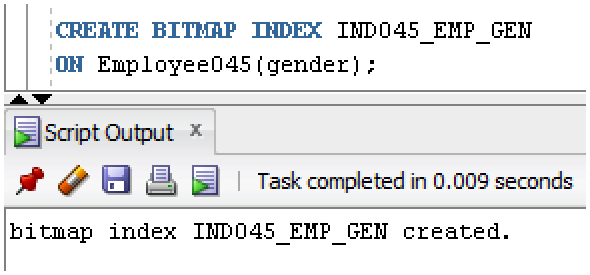
```sql
CREATE INDEX Training045_title10_idx
ON Training045(SUBSTR(title, 1, 10));
```

Task completed in 0.009 seconds

```
index TRAINING045_TITLE10_IDX created.
```

3. Create **bitmapped** indexes on

i) **Gender** column of EMP table

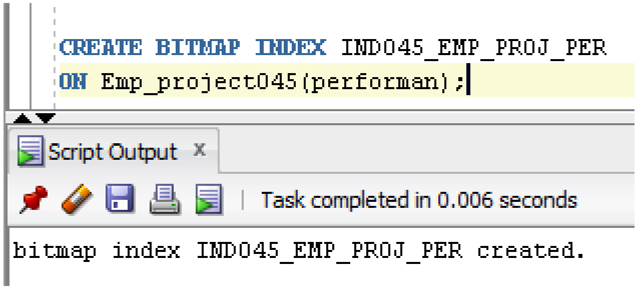


```sql
CREATE BITMAP INDEX IND045_EMP_GEN
ON Employee045(gender);
```

Task completed in 0.009 seconds

```
bitmap index IND045_EMP_GEN created.
```

ii) **Performance** column of EMP_PROJECT table



```sql
CREATE BITMAP INDEX IND045_EMP_PROJ_PER
ON Emp_project045(performan);
```

Task completed in 0.006 seconds

```
bitmap index IND045_EMP_PROJ_PER created.
```