Create Sequences, Indexes and synonyms

Lab session 09

SEQUENCE

SEQUENCE

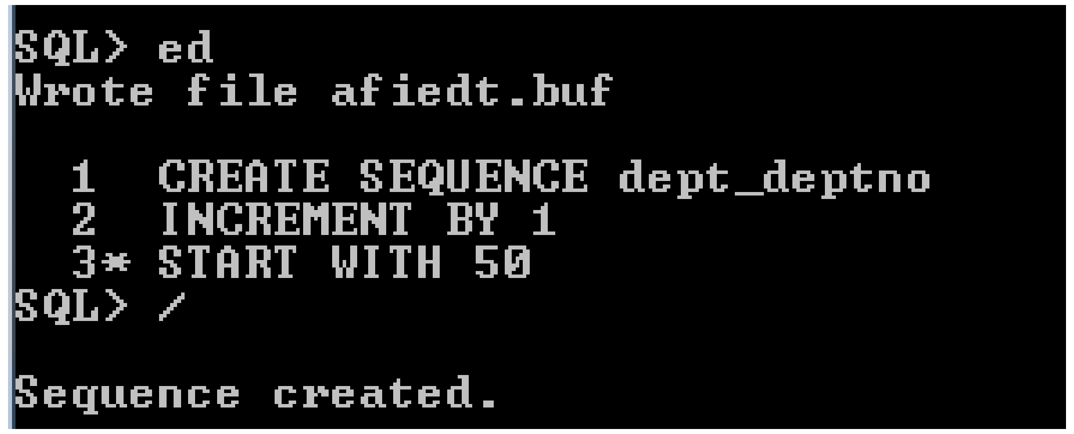
- ➢Sequence generator can be used to automatically generate sequence numbers for rows in tables.
- ➢A sequence is a database object created by a user and can be shared by multiple users.
- ➢A typical usage for sequences is to create a primary key value, which must be unique for each row.
- ➢The sequence 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.

● Syntax of Sequence

```
CREATE SEQUENCE Sequence_Name
[INCREMENT BY n]
[START WITH n];
```

Example: create a sequence DEPT_DEPTNO that can be use as a primary key of department table.

```
CREATE SEQUENCE dept_deptno
INCREMENT BY 1
START WITH 50
```



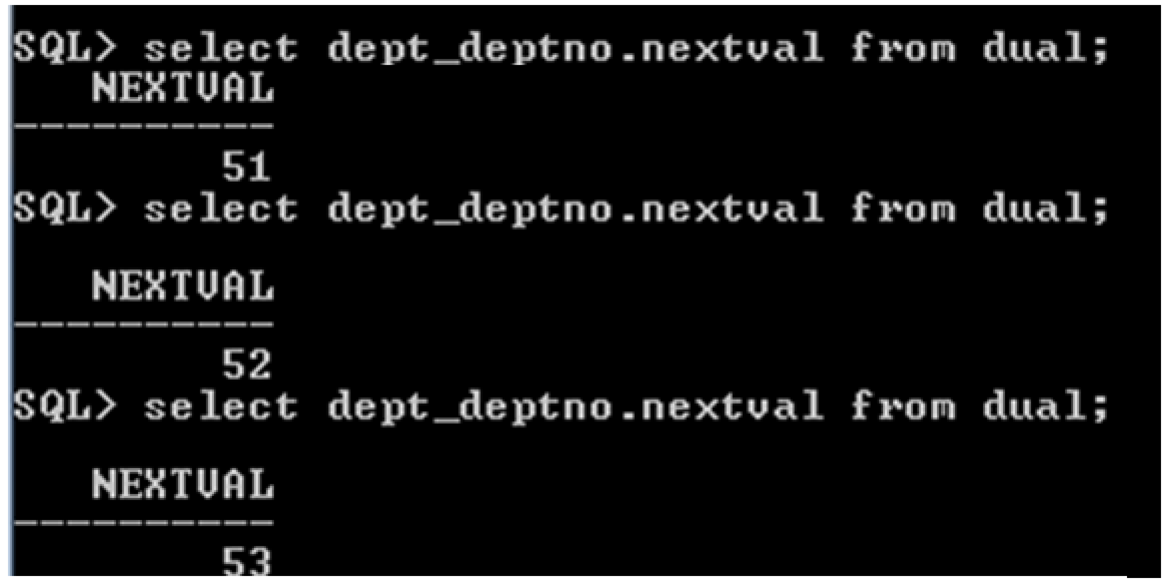
```
SQL> ed
Wrote file afiedt.buf

  1  CREATE SEQUENCE dept_deptno
  2  INCREMENT BY 1
  3* START WITH 50
SQL> /

Sequence created.
```

SEQUENCE

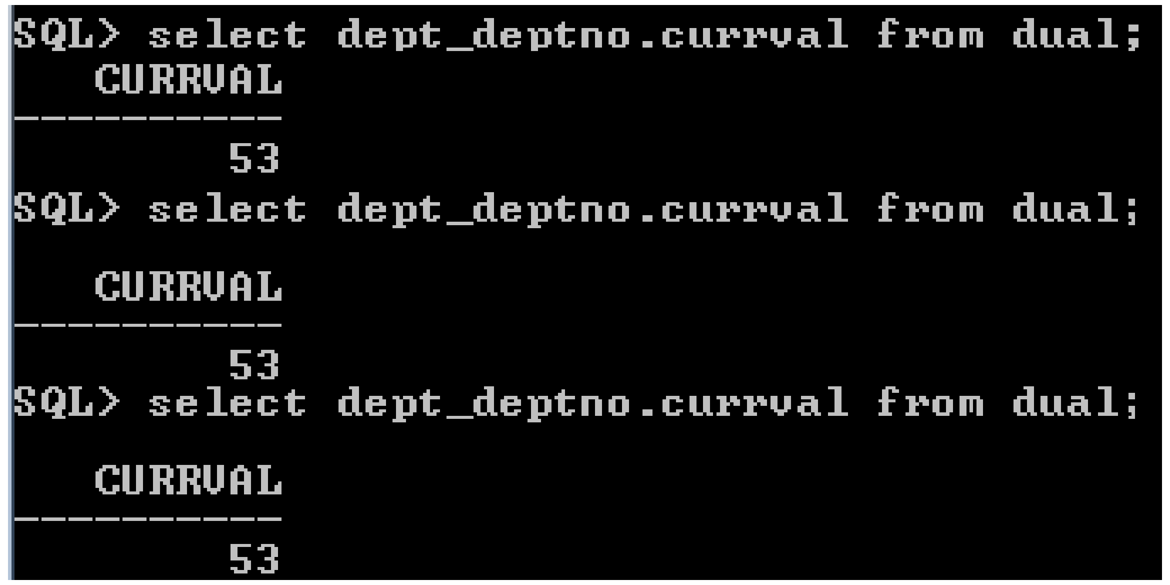
Using a sequence

Using a sequence we must know **NEXTVAL** and **CURRVAL** pseudo columns

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



```
SQL> select dept_deptno.nextval from dual;
   NEXTVAL
----------
        51
SQL> select dept_deptno.nextval from dual;

   NEXTVAL
----------
        52
SQL> select dept_deptno.nextval from dual;

   NEXTVAL
----------
        53
```

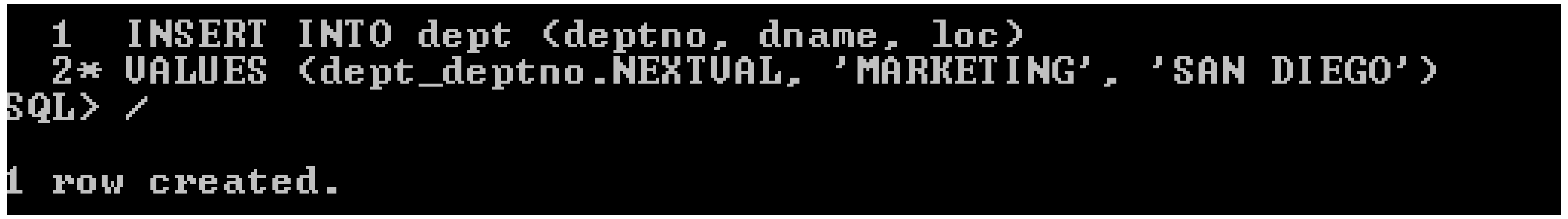


```
SQL> select dept_deptno.currval from dual;
   CURRVAL
----------
        53
SQL> select dept_deptno.currval from dual;

   CURRVAL
----------
        53
SQL> select dept_deptno.currval from dual;

   CURRVAL
----------
        53
```

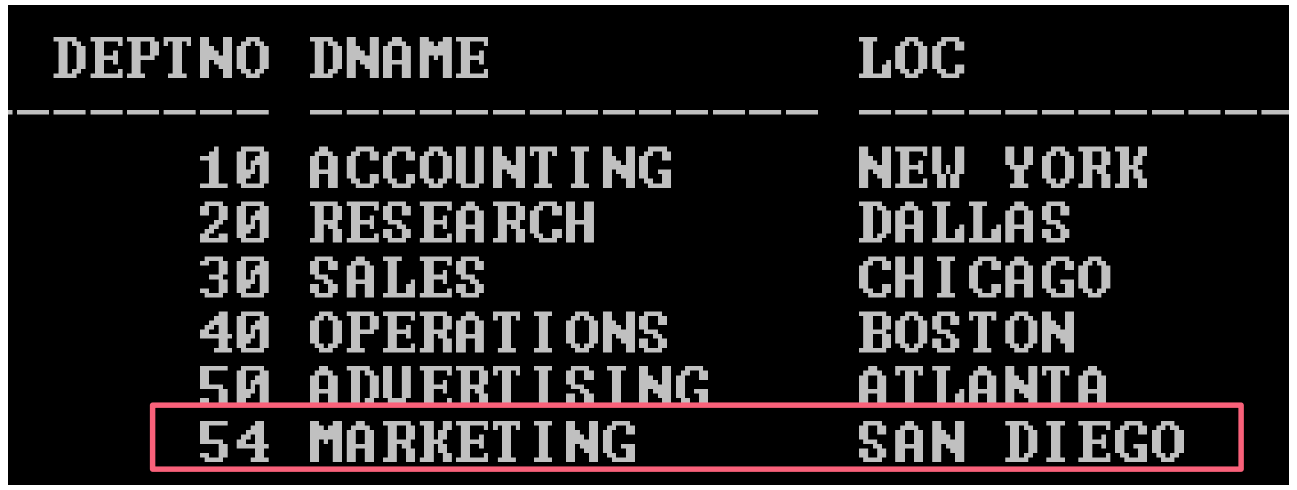
CURRVAL returns the current sequence value. That could be updated or incremented by using **NEXTVAL**

Example: create a new department named as *MARKETING* in *San Diego* by using sequence **dept_deptno** we generated above.

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



```
  1  INSERT INTO dept (deptno, dname, loc)
  2* VALUES (dept_deptno.NEXTVAL, 'MARKETING', 'SAN DIEGO')
SQL> /

1 row created.
```

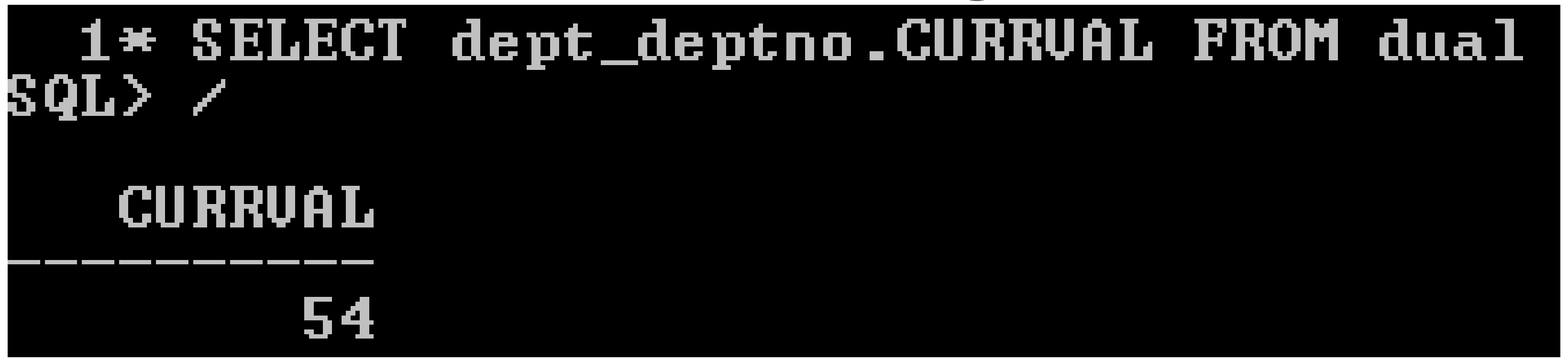


```
 DEPTNO DNAME          LOC
------- -------------- -------------
     10 ACCOUNTING     NEW YORK
     20 RESEARCH       DALLAS
     30 SALES          CHICAGO
     40 OPERATIONS     BOSTON
     50 ADVERTISING    ATLANTA
     54 MARKETING      SAN DIEGO
```

SEQUENCE

● Using a sequence(cont)

Now lets check current value again



```
  1* SELECT dept_deptno.CURRVAL FROM dual
SQL> /

   CURRVAL
----------
        54
```

● Removing a sequence

Sequence can be removed by following statement:

DROP SEQUENCE sequence_name

INDEXES

INDEXES

➢index is a schema object that can speed up the retrieval of rows by using a pointer
➢Indexes can be created explicitly or automatically(Oracle automatically creates an index for each UNIQUE or PRIMARY KEY declaration unless there is no other index present on that column).
➢An index provides direct and fast access to rows in a table and hence reduce the unnecessary disk I/O operations by using an indexed path to locate data quickly.
➢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.
➢Oracle maintains the indexes automatically. when new rows are added to the table, updated, or deleted, Oracle updates the corresponding indexes.

● Types of indexes

We can create the following indexes:-

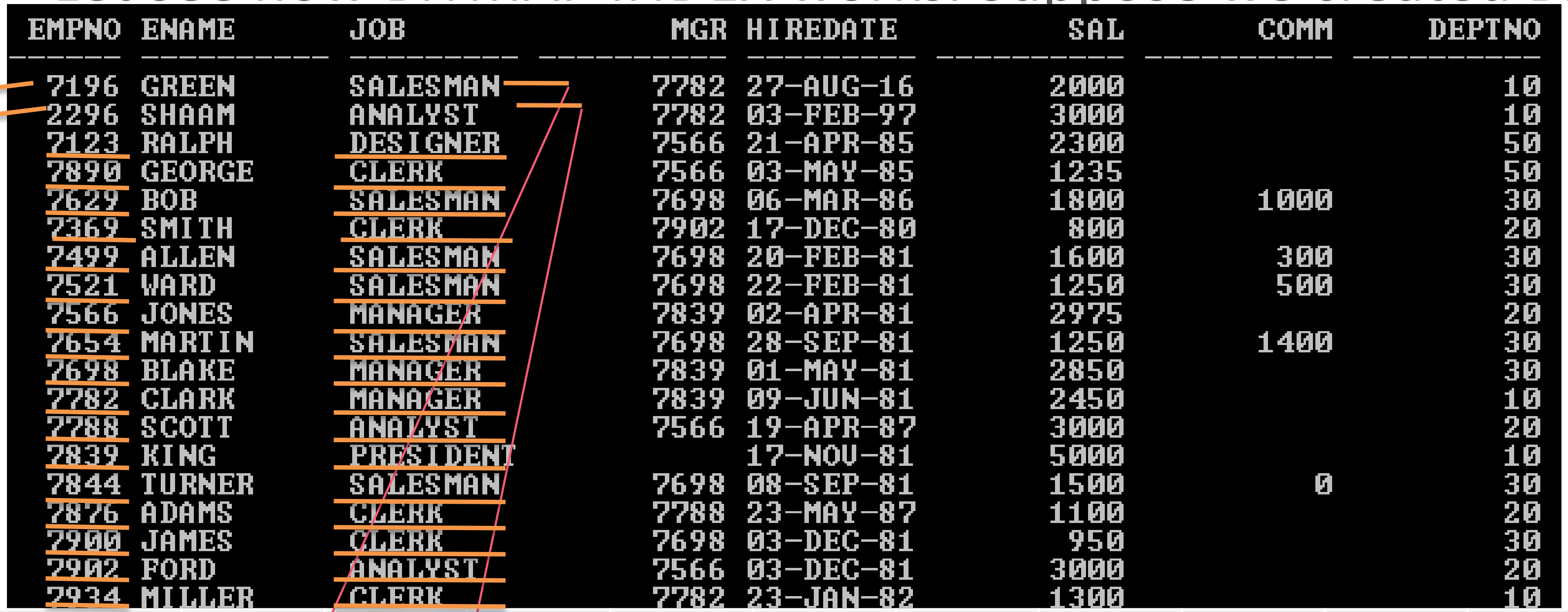
BITMAP INDEX 1

➢Bitmap index does not repeatedly stores the index column values means it stores a bit(0 or 1) against the corresponding ROW IDs and their respective values(maintains a 2D array).
What is Cardinality?
Cardinality is the number of distinct column values in a particular column.
➢Bitmap index is suitable for columns with low cardinality such as Gender column where possible values are **Male** and **Female** (cardinality is 2)other wise BITMAP index would be less effective than a normal search.

INDEXES

BITMAP INDEX(cont) 1

➢Let see how BITMAP INDEX works. Suppose we created Bitmap index on **JOB** column of **EMP** table



```
 EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
------ ---------- --------- ---------- --------- ---------- ---------- ----------
  7196 GREEN      SALESMAN        7782 27-AUG-16       2000                    10
  2296 SHAAM      ANALYST         7782 03-FEB-97       3000                    10
  7123 RALPH      DESIGNER        7566 21-APR-85       2300                    50
  7890 GEORGE     CLERK           7566 03-MAY-85       1235                    50
  7629 BOB        SALESMAN        7698 06-MAR-86       1800       1000         30
  7369 SMITH      CLERK           7902 17-DEC-80        800                    20
  7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
  7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
  7566 JONES      MANAGER         7839 02-APR-81       2975                    20
  7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
  7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
  7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
  7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
  7839 KING       PRESIDENT            17-NOV-81       5000                    10
  7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
  7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
  7900 JAMES      CLERK           7698 03-DEC-81        950                    30
  7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
  7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
```

So Distinct values in JOB columns are: ***SALESMAN*,*ANALYST*,*DESIGNER*,*CLERK*,*MANAGER*** & ***PRESIDENT***

These distinct values will store in a column of BITMAP Index(a 2D array).

	7196	*2296*	*7123*	*7890*	*7629*	*7369*	*7499*	*7521*	*7566*	*7654*	*7698*	*7782*	*7788*	*7839*	*7844*	*7876*	*7900*	*7902*	*7934*
SALESMAN	1	0	0	0	1	0	1	1	0	1	0	0	0	0	1	0	0	0	0
ANALYST	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
DESIGNER	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CLERK	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1
MANAGER	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0
ANALYST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRESIDENT	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

While the rest bits are Zero in Bitmap index

INDEXES

BITMAP INDEX (Cont) 1

➢Searching would be advance in such a way that to get all analyst the IDs against the ones in ***Analyst row*** in Bitmap will be returned

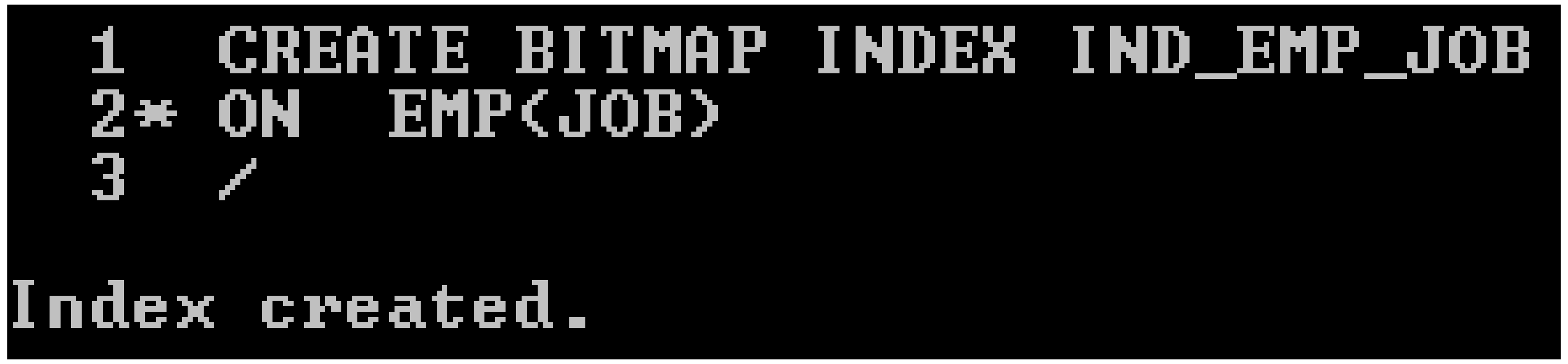
➢You can only see the practical benefit of Index when there would be billions and trillion of records

Creating BITMAP INDEX

```
CREATE BITMAP INDEX INDEX_NAME
ON  TABLENAME(COLUMN)
```

Creating BITMAP INDEX on Emp Table's JOB column

```
CREATE BITMAP INDEX IND_EMP_JOB
ON  EMP(JOB)
```



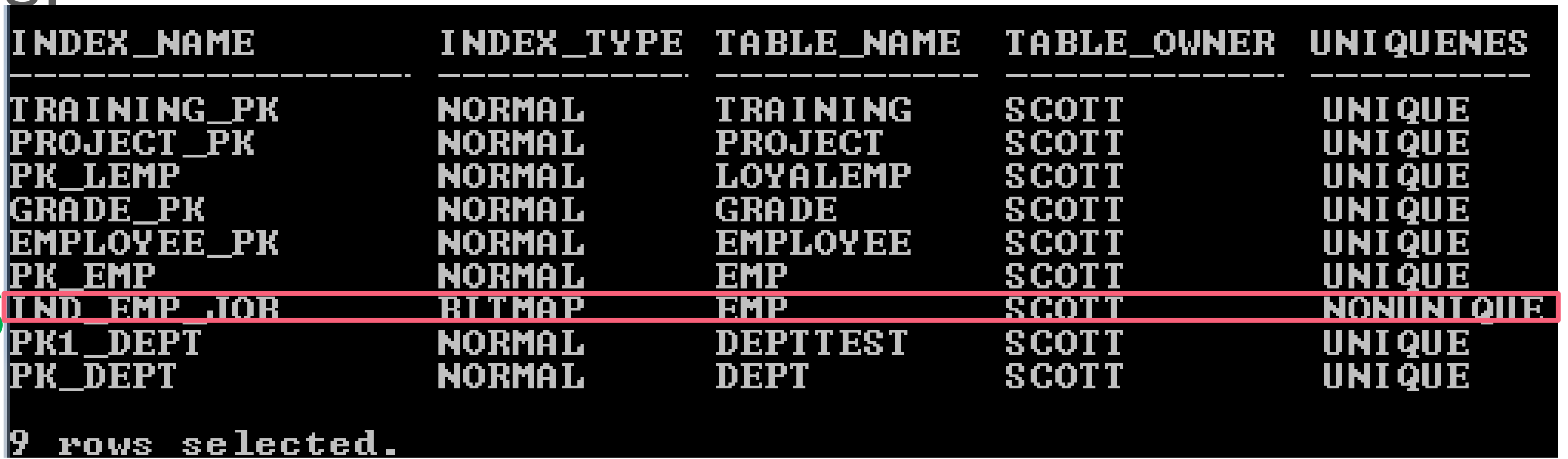
```
  1   CREATE BITMAP INDEX IND_EMP_JOB
  2*  ON   EMP(JOB)
  3   /

Index created.
```

Confirming INDEX:

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

Set Wrap off

```
SELECT INDEX_NAME,INDEX_TYPE,
TABLE_NAME, TABLE_OWNER, UNIQUENESS
FROM USER_INDEXES
```

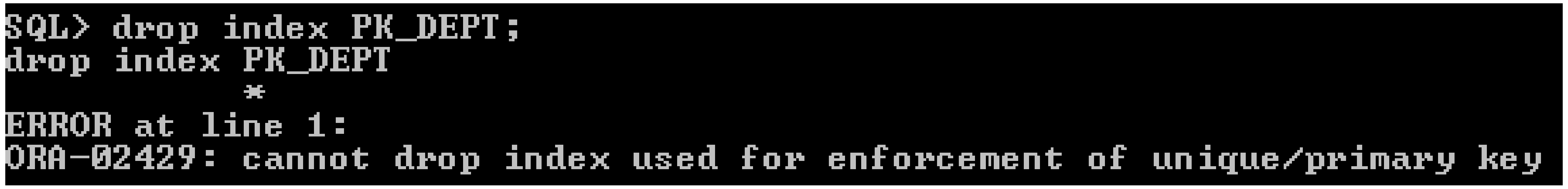


```
INDEX_NAME                 INDEX_TYPE  TABLE_NAME   TABLE_OWNER  UNIQUENES
-------------------------- ----------- ------------ ------------ ---------
TRAINING_PK                NORMAL      TRAINING     SCOTT         UNIQUE
PROJECT_PK                 NORMAL      PROJECT      SCOTT         UNIQUE
PK_LEMP                    NORMAL      LOYALEMP     SCOTT         UNIQUE
GRADE_PK                   NORMAL      GRADE        SCOTT         UNIQUE
EMPLOYEE_PK                NORMAL      EMPLOYEE     SCOTT         UNIQUE
PK_EMP                     NORMAL      EMP          SCOTT         UNIQUE
IND_EMP_JOB                BITMAP      EMP          SCOTT         NONUNIQUE
PK1_DEPT                   NORMAL      DEPTTEST     SCOTT         UNIQUE
PK_DEPT                    NORMAL      DEPT         SCOTT         UNIQUE

9 rows selected.
```

I N D E X E S

B-TREE INDEX 2

- Searching is done using **B**inary **S**earch **T**ree. Do you know BST? ***video***
- This is the default index means we don't need to write the name of **B-TREE** Index when creating Index.
- This index is created using B Tree Algoritm.
- The b-tree includes nodes with the index column values and the ROWIDs (used to identify the rows in the table)

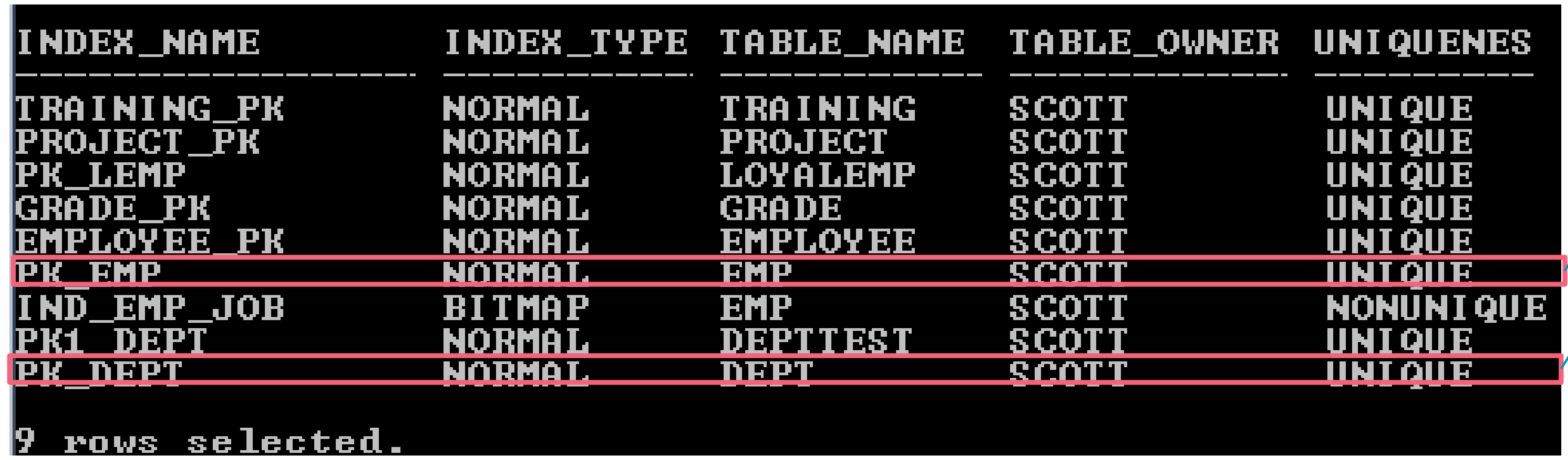


```
SQL> drop index PK_DEPT;
drop index PK_DEPT
           *
ERROR at line 1:
ORA-02429: cannot drop index used for enforcement of unique/primary key
```

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 constraint. You cant even Drop them.



```
INDEX_NAME         INDEX_TYPE TABLE_NAME  TABLE_OWNER UNIQUENES
------------------ ---------- ----------- ----------- ---------
TRAINING_PK        NORMAL     TRAINING    SCOTT       UNIQUE
PROJECT_PK         NORMAL     PROJECT     SCOTT       UNIQUE
PK_LEMP            NORMAL     LOYALEMP    SCOTT       UNIQUE
GRADE_PK           NORMAL     GRADE       SCOTT       UNIQUE
EMPLOYEE_PK        NORMAL     EMPLOYEE    SCOTT       UNIQUE
PK_EMP             NORMAL     EMP         SCOTT       UNIQUE
IND_EMP_JOB        BITMAP     EMP         SCOTT       NONUNIQUE
PK1_DEPT           NORMAL     DEPTTEST    SCOTT       UNIQUE
PK_DEPT            NORMAL     DEPT        SCOTT       UNIQUE

9 rows selected.
```

Indexes created as soon as Primary Keys have been defined

Non Unique Index

Users can create non-unique 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.***(foreign key of dept is not in any order in emp table)***

I N D E X E S

B-TREE INDEX 2

Non Unique Index(cont)

Unique index is define as unique so all associated index entries must be unique. It's simply not possible to have duplicate index entries within a Unique index structure **index column** which uniquely identify each row. Therefore, it's not necessary to have the rowid as **a separate column** of the index entry in **Unique Index**.

As far as Non Unique index is concern it stores **rowid** with separate **index column**. **Now,** If we will delete and re-insert the same index value **within a single transaction** then in unique index case one row would be deleted and same row would be inserted again but as far as **non unique index** is concern if we need to delete and re-insert the same index value **within a single transaction**, Oracle is forced to create a new index entry and will not reuse the existing one so the size of the Non Unique index structure is increasing every time. Keeping row id a side there is no uniqueness in the index that's why called as Non Unique index.

Function Based Index

The function-based index can be created on columns with expressions like (SAL + COMM) and SUBSTR(EMPID,1,2).
Pre-store the computed values is the primary objective of Function based Index.

I N D E X E S

Function Based Index(cont)

EXAMPLE 1 :Creating an index on SUBSTR(EMPNO,1,2) will advance your search in the following way

Let say I have 10000 employees in my company & I need to search a empno 7834 so if I made index as mentioned in above example **SUBSTR(EMPNO,1,2)** so record would only be searched under those employees list who have first starting 2 digits as 78.

EMP ID	EMP NAME
77	ALI
78	DANIAL
78	HADIQA
78	SHADAB
78	SAAD
79	KHUBAIB
79	SAMRA

Traversing only these record during a search. Assuming a small chunk of data here from whole data

```
CREATE INDEX EMP_IDX_NEW
ON  EMP(SUBSTR(EMPNO,1,2))
```

As Btree is By default so Just write INDEX.

EXAMPLE 2 :Let say we have a table TARGET_EMPLOYEE AS FOLLOWS:

EMPNO	SALARY	LOAN	COMPANY EXPENSES
7874	1000	50000	2000
7875	50000	0	0
7824	100000	200000	50000
7893	40000	20000	1000

I N D E X E S

Function Based Index(cont)

EXAMPLE 2 :Let say we have a table TARGET_EMPLOYEE AS FOLLOWS:

Assuming a small chunk of data here from whole data

EMPNO	SALARY	LOAN	COMPANY_ EXPENSES
7874	1000	50000	2000
7875	50000	0	0
7824	100000	200000	50000
7893	40000	20000	1000

54
92
112
333

Assuming Row IDS

Suppose company need to fire those employees who have salary + Expense + Company expense >**300000.** without index if we will have billions and trillions of records in our employee table it will take time to retrieve record with calculation so index will be maintained as

SAL + LOAN + CE	ROW ID
53000	54
50000	92
350000	112
61000	333

This record will be Targeted record

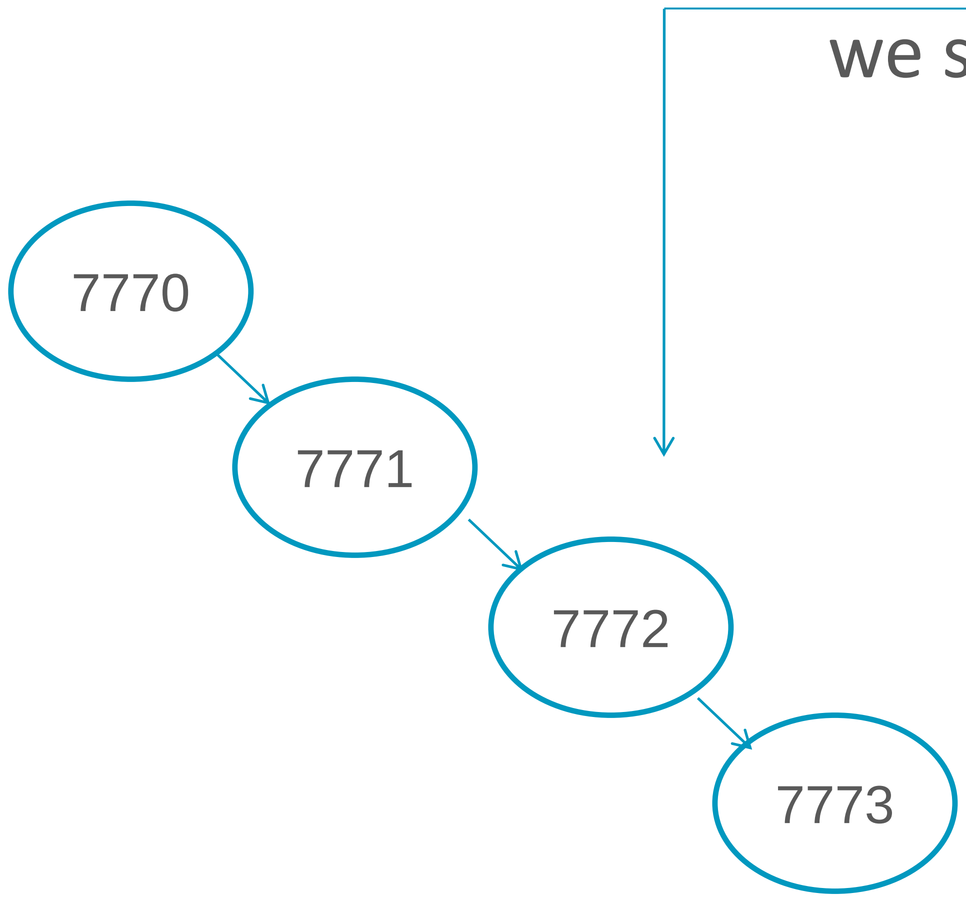
```
CREATE INDEX EMP_TARGET_IDX
ON  TARGET_EMPLOYEE(SAL + LOAN + COMPANY_EXPENSES)
```

INDEXES

Reverse Key Index

This index is beneficial in Oracle parallel server environment other wise where at a time concurrent inserts can be performed and thus concurrent searches can also be performed. But if a particular **memory block** is acquired by one process of Oracle server then other process of either search or insert have to wait for that **memory block.** This issue is called as **index block contention.**

as Primary Keys generated by a sequence so will definitely result to contention as all inserts need to access the maximum "right-most" leaf block(greater one according to Binary tree we saw in video). Let say in my data I am inserting ids 7771,then 7772,then 7773 and so on



So inserting is a one time process but as far as search is concern and particularly a parallel search is concern then in parallel server environment if values will be present in sequence then every **search process** have to wait for another **search process** who acquired a respective block of memory as data is present in a sequence. In **Reverse Key Index** We save ids after breaking their sequence by reversing their values means 7771 will be save in index as 1777 and 7772 will be save in index as 2777 and 7773 will be save in index as 3777 and so on.

Note: Students you are smart intelligent so you must know that Reverse Key Indexes address this specific problem but may in turn introduce a number of problems due to unsorted data.

INDEXES

CLUSTER INDEXES	NON CLUSTER INDEXES
A **clustered index** actually describes the order in which records are physically stored on the disk, hence the reason you can only have one.	A **Non-Clustered Index** defines a logical order that does not match the physical order on disk.
Normally Unique index is a cluster index	Reverse key index can be a non cluster index.

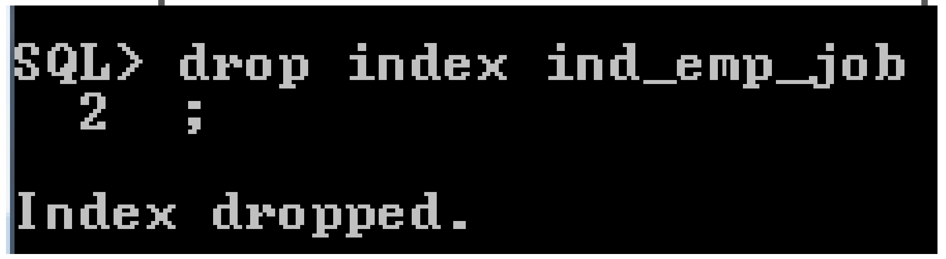
Removing an Index

It is not possible to modify an index. To change it, we must drop it first and then re-create it.

Remove an index we use:
DROP INDEX *index_name;*

Remove an index of any user we write:
DROP INDEX **ANY** *index_name;*

Example: we created a bitmap index IND_EMP_JOB. lets remove it



```
SQL> drop index ind_emp_job
  2  ;

Index dropped.
```

SYNONYMS

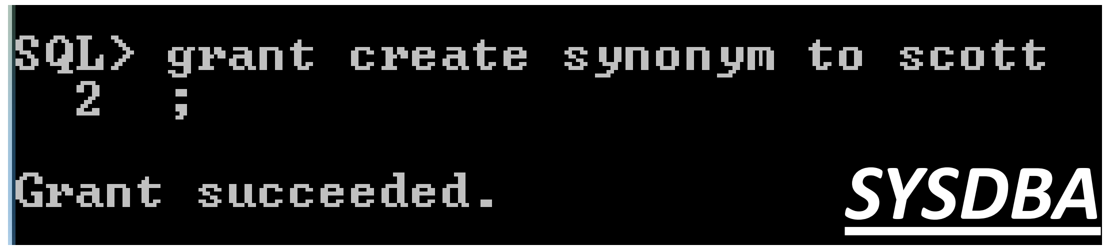
S Y N O N Y M S

A **synonym** is an alias or alternate name for a ***table***, ***view***, ***sequence***, or other ***Database objects***. They are used mainly to make it easy for users to access **database** objects owned by other users. They hide the underlying object's identity and make it harder for a malicious program or user to target the underlying object.

Syntax of Synonyms

CREATE [PUBLIC] SYNONYM *synonym_name*
FOR *object;*

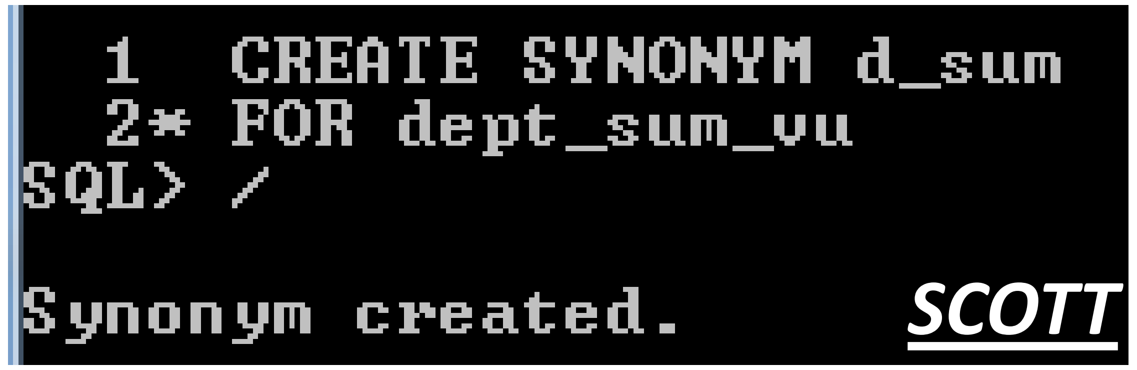
Example :To create a synonym the DEPT_SUM_VU view we create in last lab(lab 08) (before that u need privilges for creating synonyms via **sysdba** to **scott** user)



```
SQL> grant create synonym to scott
  2  ;

Grant succeeded.
```

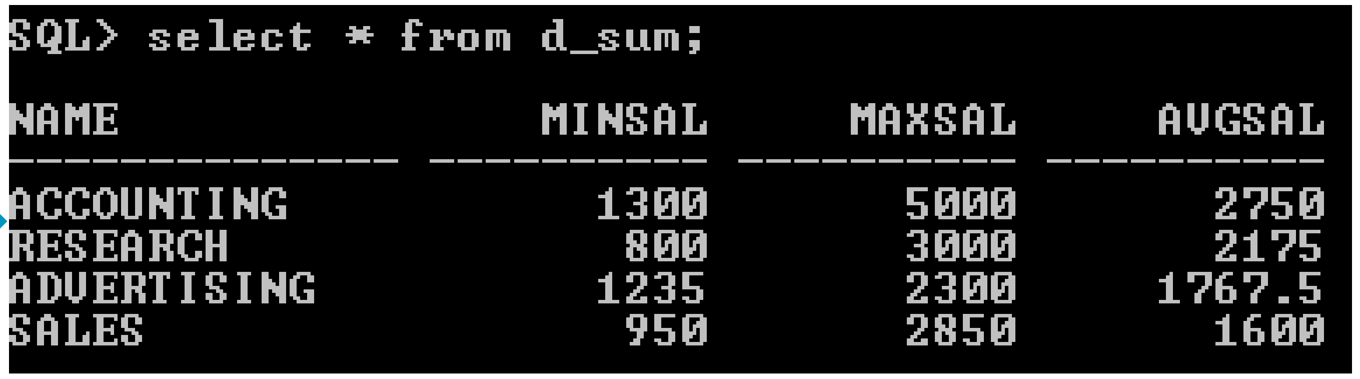
SYSDBA

CREATE SYNONYM *d_sum*
FOR *dept_sum_vu*;



```
  1  CREATE SYNONYM d_sum
  2* FOR dept_sum_vu
SQL> /

Synonym created.
```

SCOTT

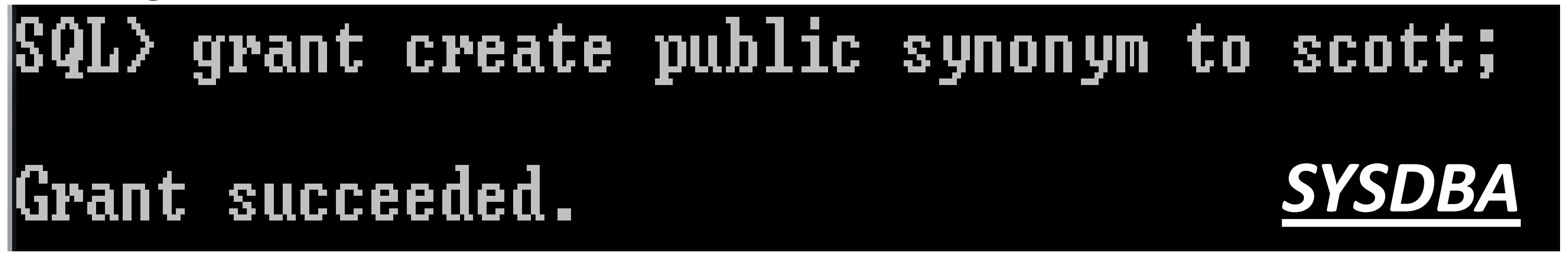


```
SQL> select * from d_sum;

NAME                MINSAL     MAXSAL     AVGSAL
--------------- ---------- ---------- ----------
ACCOUNTING            1300       5000       2750
RESEARCH               800       3000       2175
ADVERTISING           1235       2300     1767.5
SALES                  950       2850       1600
```

S Y N O N Y M S

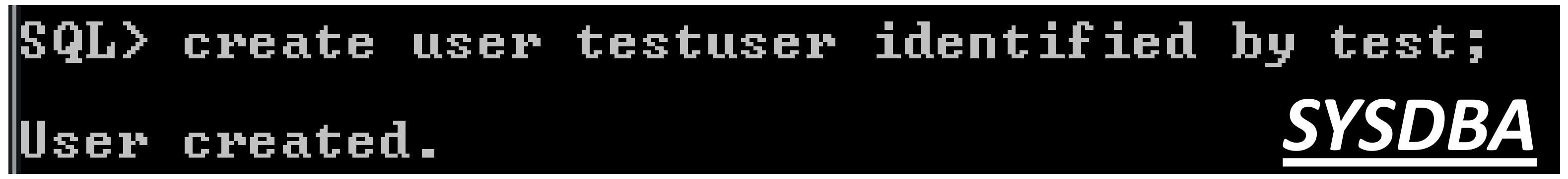
Example :create a public synonym named ***DEPT*** for SCOTT's DEPT table:

NOTE :to create public synonyms you need to give rights to via Sysdba user for **CREATE PUBLIC SYNONYMS** to **scott**



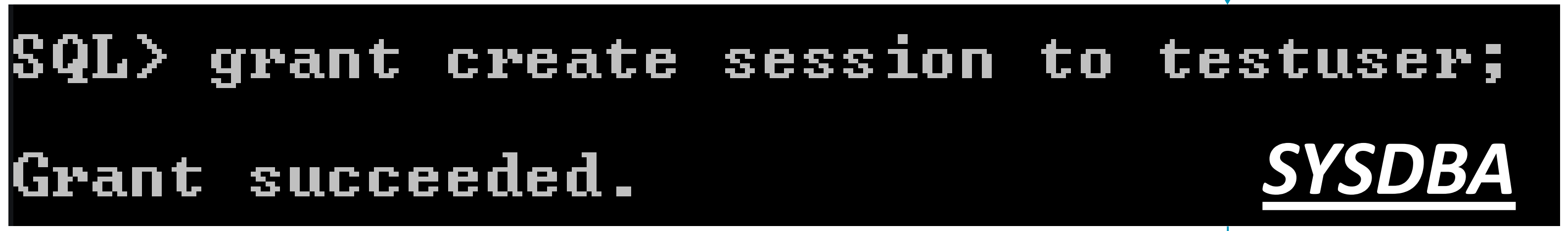
```
SQL> grant create public synonym to scott;

Grant succeeded.
```

To test this scenaro let us create a **testuser** with **test** as password via **sysdba.**

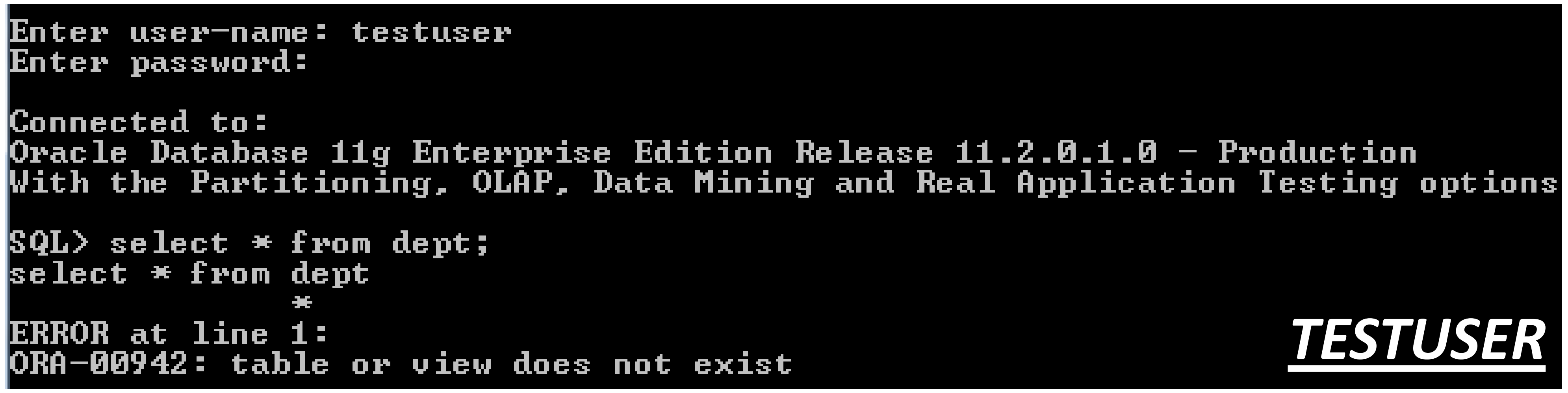


```
SQL> create user testuser identified by test;

User created.
```

Before logon to **testuser** we need **session** right also via **sysdba** to **testuser**



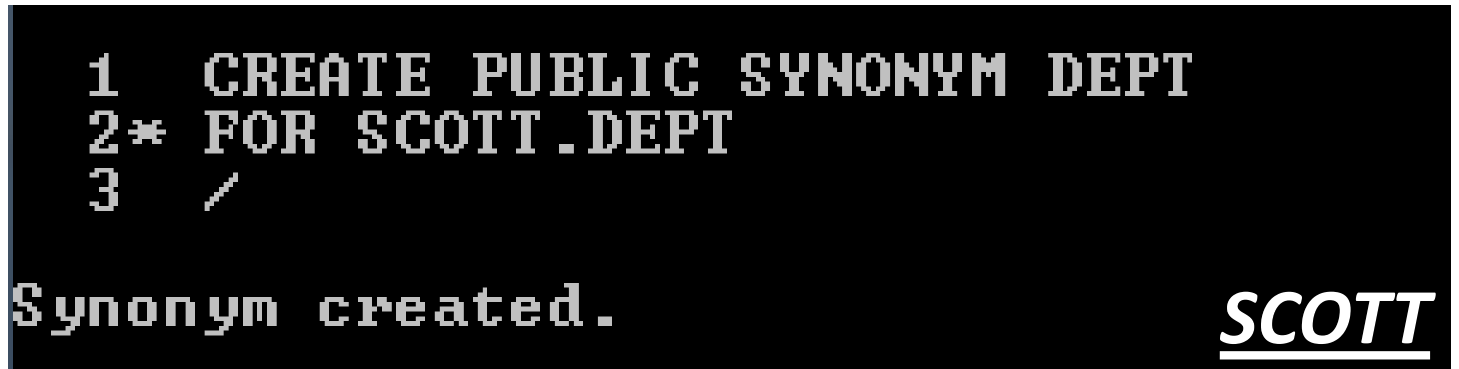
```
SQL> grant create session to testuser;

Grant succeeded.
```



```
Enter user-name: testuser
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> select * from dept;
select * from dept
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Lets create public SYNONYM in ***scott*** for department table so that it can be accessed via ***testuser***

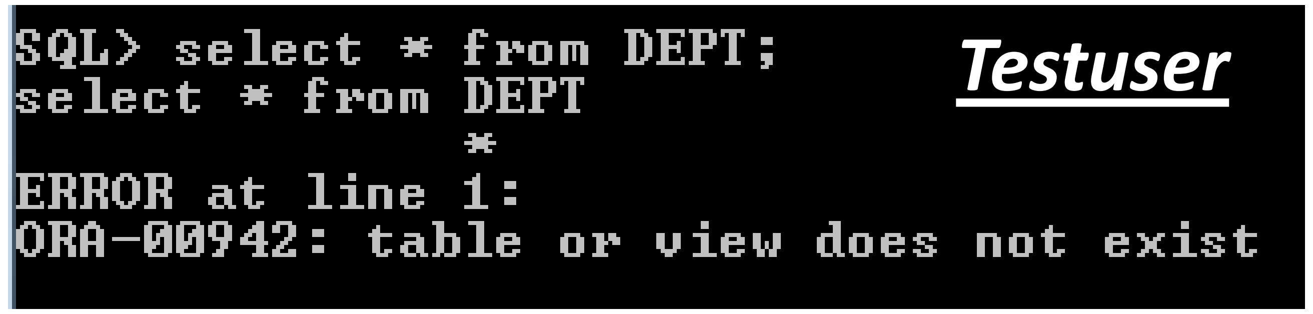
CREATE PUBLIC SYNONYM DEPT
FOR SCOTT.DEPT;



```
  1  CREATE PUBLIC SYNONYM DEPT
  2* FOR SCOTT.DEPT
  3  /

Synonym created.
```

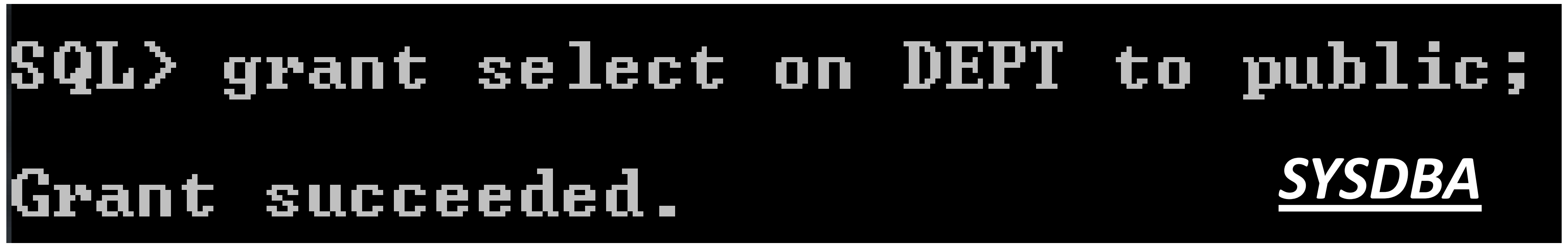
SYNONYMS

Now lets access the PUBLIC SYNONYM DEPT from testuser



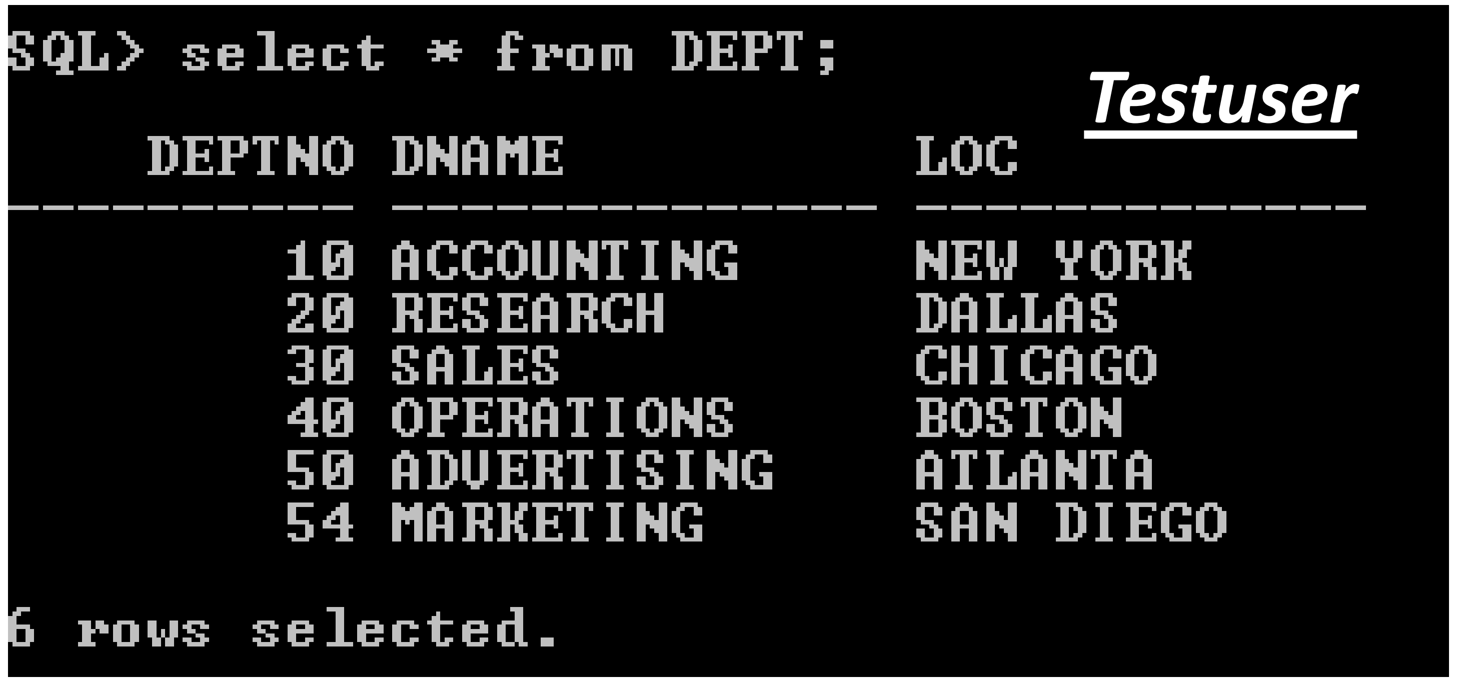
```
SQL> select * from DEPT;
select * from DEPT
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Yes you cannot access the public synonym unless a **select** grant must be given to a object(dept in our case)



```
SQL> grant select on DEPT to public;

Grant succeeded.
```

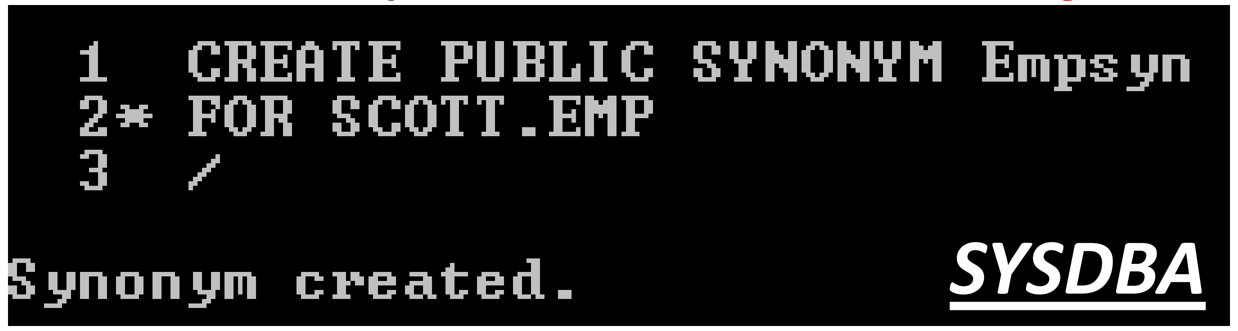
Now lets again access the PUBLIC SYNONYM DEPT from testuser



```
SQL> select * from DEPT;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 ADVERTISING    ATLANTA
        54 MARKETING      SAN DIEGO

6 rows selected.
```

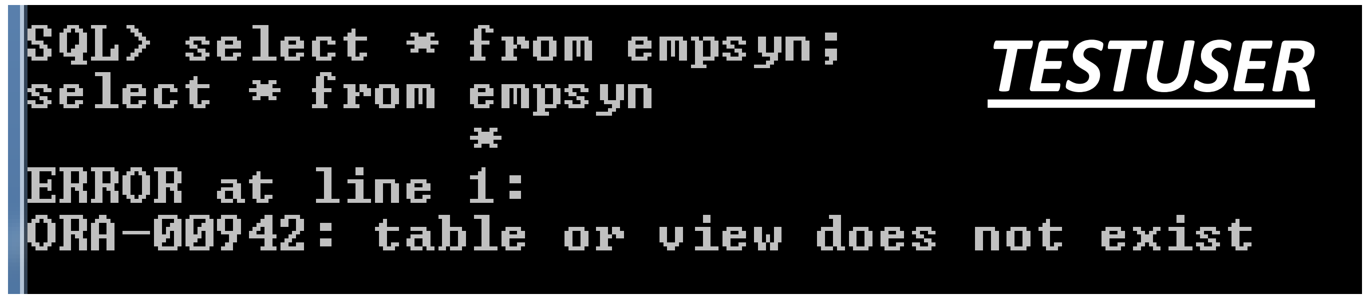
SYNONYMS

Lets create public SYNONYM in ***sysdba*** for Employee table of ***sysdba*** so that it can be accessed via ***testuser***



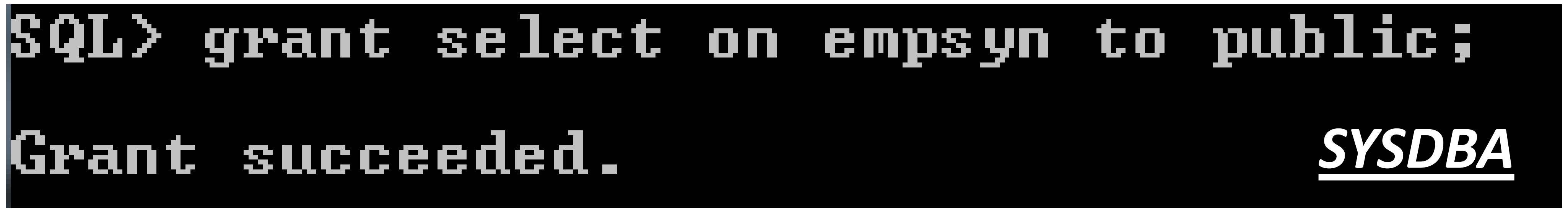
```
  1  CREATE PUBLIC SYNONYM Empsyn
  2* FOR SCOTT.EMP
  3  /

Synonym created.
```

SYSDBA



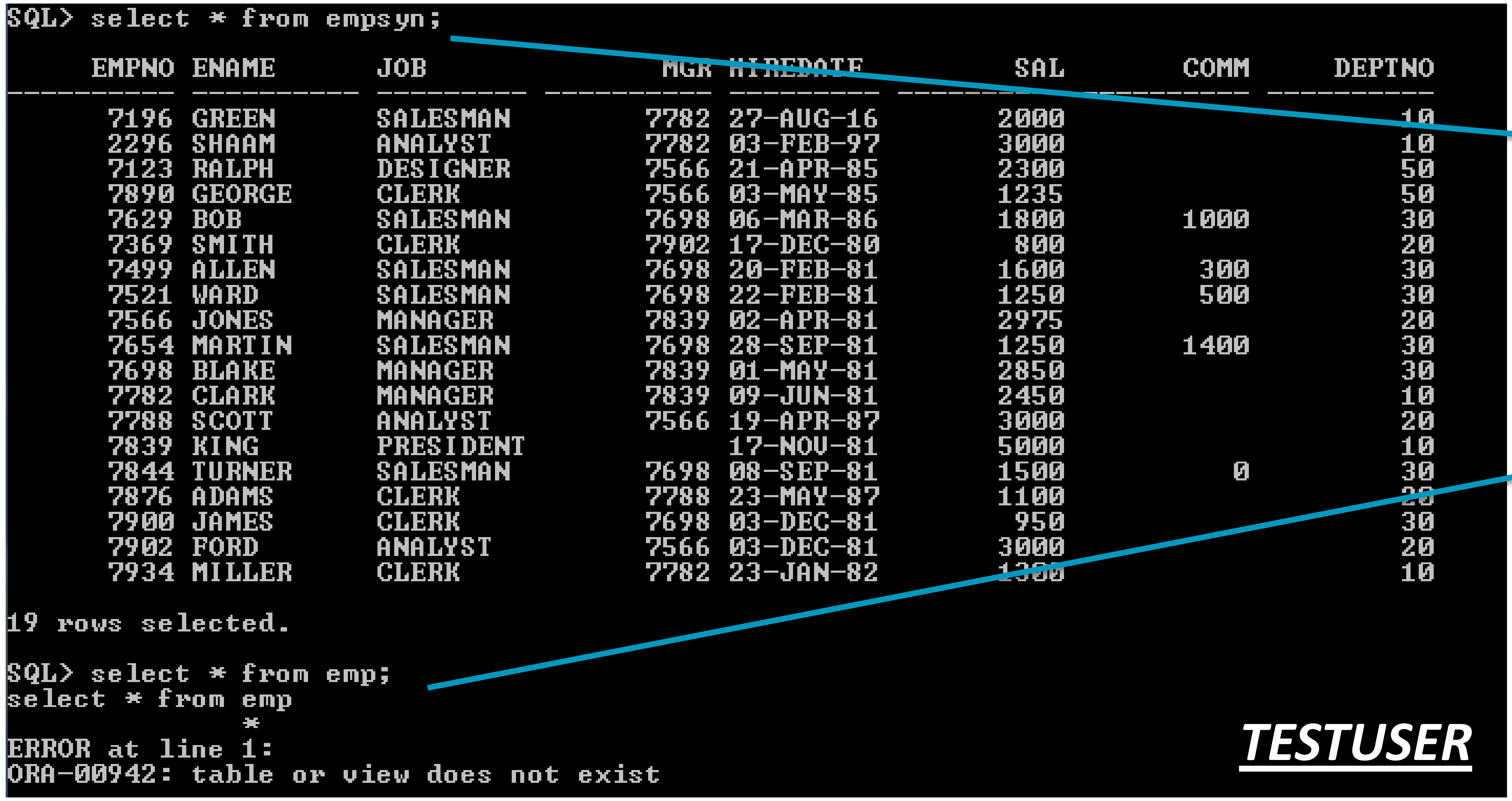
```
SQL> select * from empsyn;
select * from empsyn
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

TESTUSER



```
SQL> grant select on empsyn to public;

Grant succeeded.
```

SYSDBA



```
SQL> select * from empsyn;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7196 GREEN      SALESMAN        7782 27-AUG-16       2000                    10
      2296 SHAAM      ANALYST         7782 03-FEB-97       3000                    10
      7123 RALPH      DESIGNER        7566 21-APR-85       2300                    50
      7890 GEORGE     CLERK           7566 03-MAY-85       1235                    50
      7629 BOB        SALESMAN        7698 06-MAR-86       1800       1000         30
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

19 rows selected.

SQL> select * from emp;
select * from emp
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

TESTUSER

SYNONYM is accessing

Table is not accessing

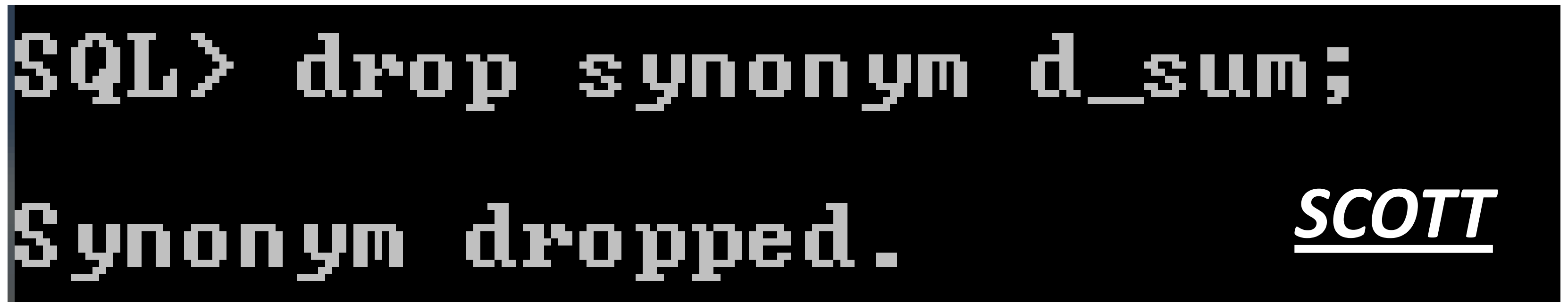
SYNONYMS

● Removing an SYNONYM

DROP SYNONYM Synonym_name;

Lets remove our all synonyms one by one

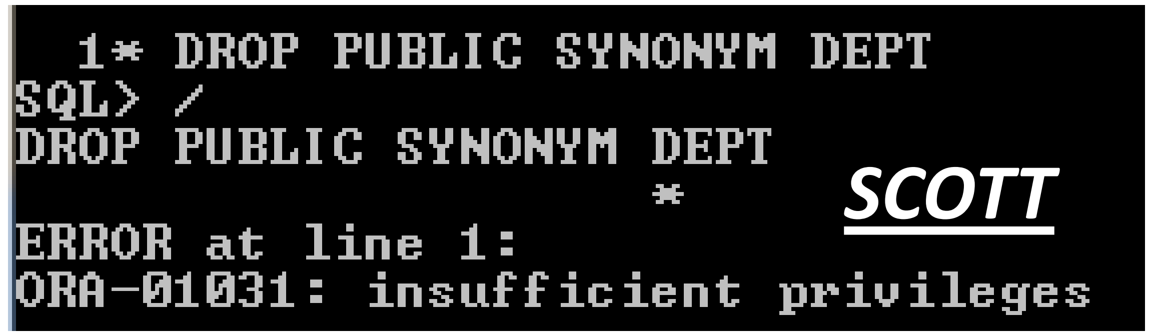
DROP SYNONYM d_sum;



```
SQL> drop synonym d_sum;

Synonym dropped.
```

SCOTT

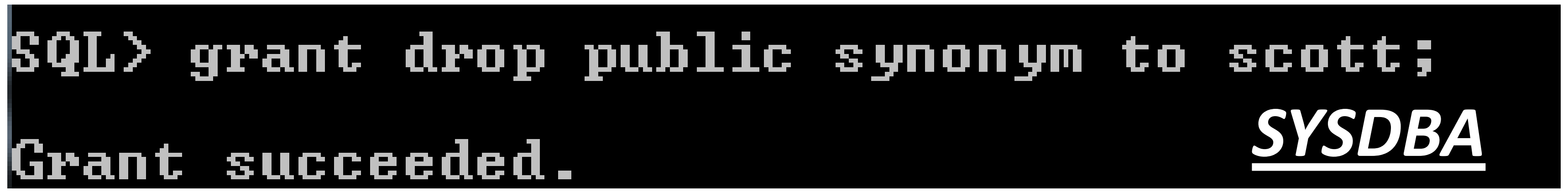
DROP PUBLIC SYNONYM DEPT;



```
  1* DROP PUBLIC SYNONYM DEPT
SQL> /
DROP PUBLIC SYNONYM DEPT
                    *
ERROR at line 1:
ORA-01031: insufficient privileges
```

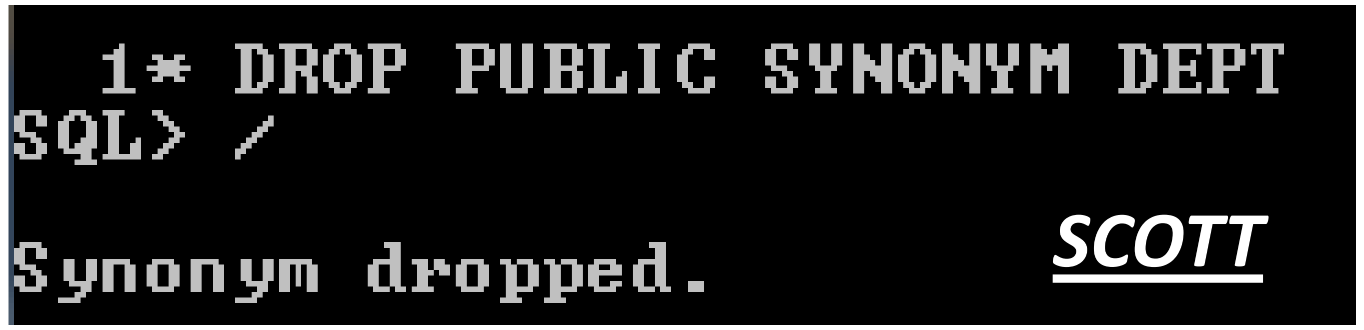
SCOTT

So we need Drop public synonym rights too



```
SQL> grant drop public synonym to scott;

Grant succeeded.
```

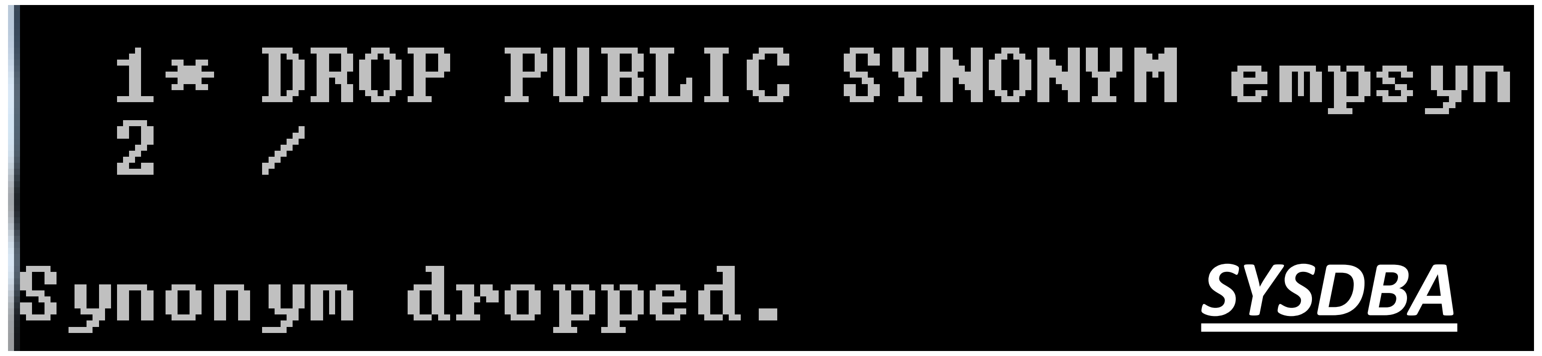
SYSDBA



```
  1* DROP PUBLIC SYNONYM DEPT
SQL> /

Synonym dropped.
```

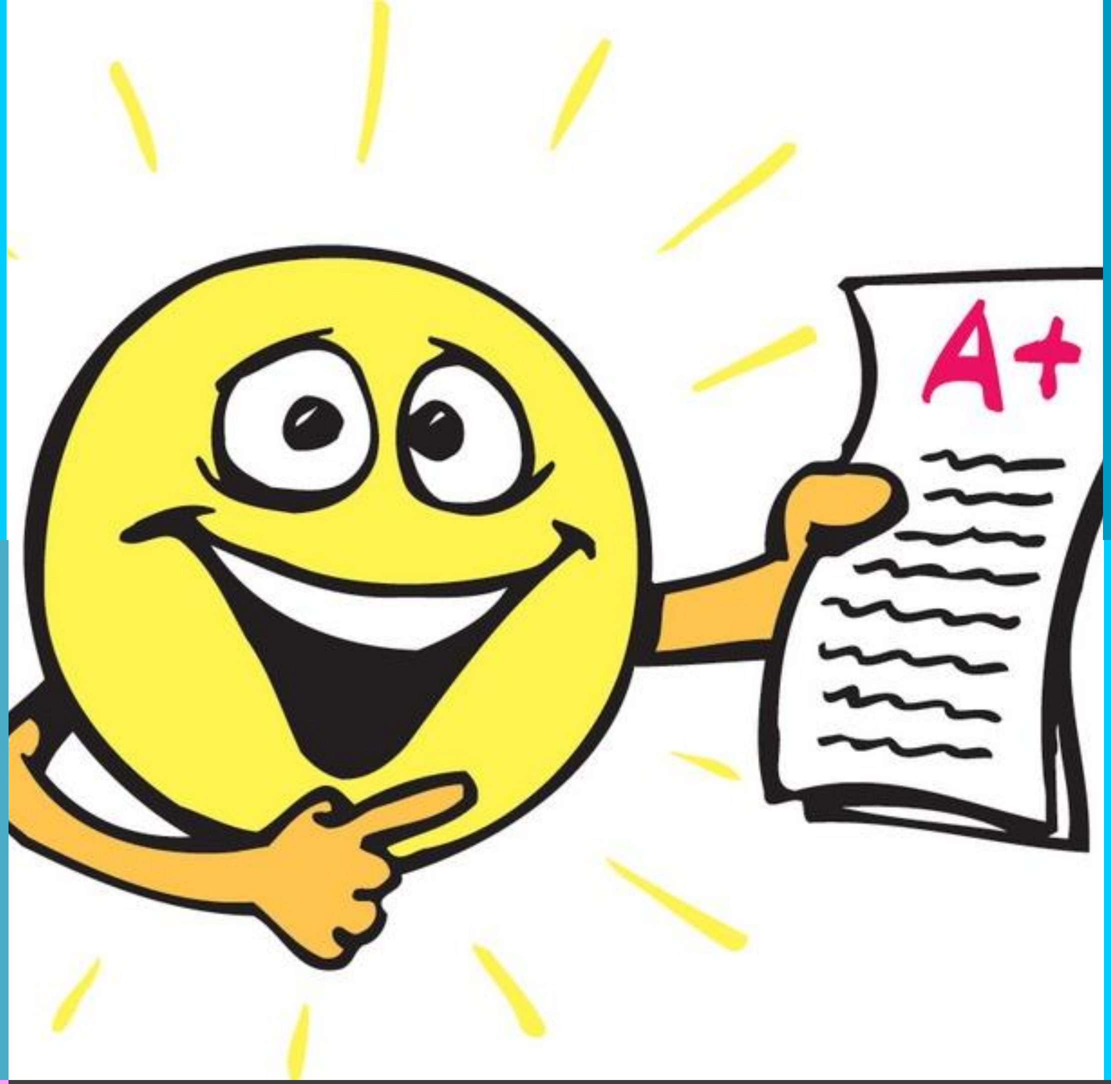
SCOTT

DROP PUBLIC SYNONYM empsyn;



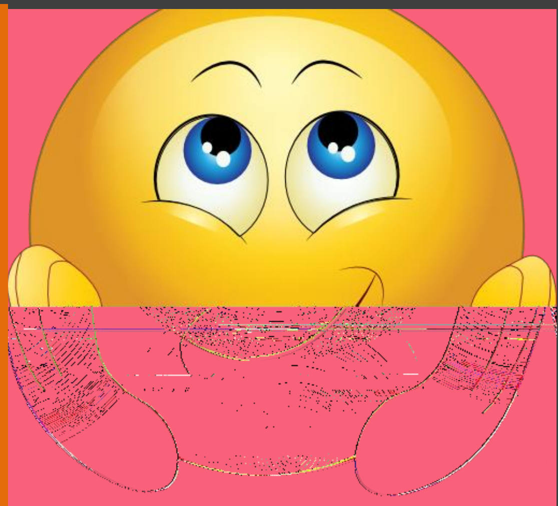
```
  1* DROP PUBLIC SYNONYM empsyn
  2  /

Synonym dropped.
```

SYSDBA



Exercise Time

Bonus Marks Time



We Have Test in next lab!

Lab 03, Lab 04 , Lab 05,Lab 06 ,Lab 07,Lab 08 & Lab 09

Practice well. This is your second test students might be the last one please prepare as best as possible.

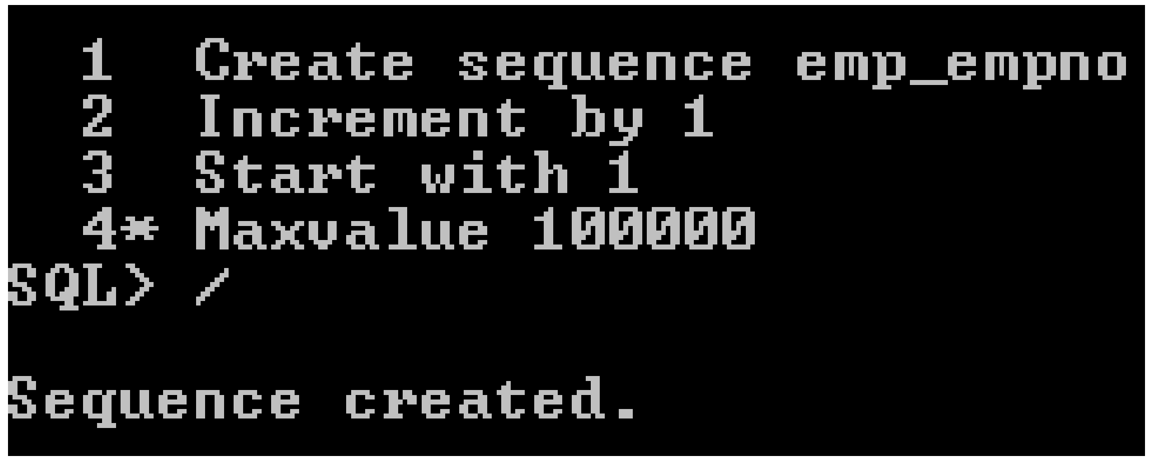
HELLO!

EXERCISE

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 emp_empno
Increment by 1
Start with 1
Maxvalue 100000;
```



```
  1  Create sequence emp_empno
  2  Increment by 1
  3  Start with 1
  4* Maxvalue 100000
SQL> /

Sequence created.
```

2. 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

Name column of EMP table

```
Create index Employee_name_idx
On EMPLOYEE(NAME);
```

Designation column of EMP table

```
Create index Employee_designation_idx
On EMPLOYEE(Designation);
```

First 10 characters of Title in TRAINING table

```
Create index Training_Title_idx
On Training(Substring(TITLE,1,10));
```

E X E R C I S E

3. Create **bitmapped** indexes on

i) **Gender** column of EMP table

ii) **Performance** column of EMP_PROJECT table

Gender column of EMP table

```
Create Bitmap index Employee_Gender_idx
On EMPLOYEE(Gender);
```

Performance column of EMP_PROJECT table

```
Create Bitmap index Employee_Project_Performance_idx
On EMPLOYEE_Project(Performance);
```

Finished