



Universidad Autónoma de Zacatecas

ACADEMIC UNIT OF ELECTRICAL ENGINEERING

Software Engineering Academic Program

Group: 5B - Semester: 2022-5^o

Practice Number: 04

Practice Name: DDL2

DATE: 08/SEPTEMBER/2022

Professor:

Aldonso Becerra Sánchez.

Student:

Cristian Omar Alvarado Rodríguez.

Índice

1. Introduction	3
2. Practice objective	4
3. Developing	4
4. Pre-assessment	29
5. Conclusion	30

DDL2

September 8th, 2022

1. Introduction

Data Definition Language (DDL) is a subset of SQL. It is a language for describing data and their relationships in a database. You can generate DDL in a database object script to:

- Keep a snapshot of the database structure.
- Set up a test system where the database acts like the production system, but contains no data.
- Produce templates for new objects that you can create based on existing ones.

DDL statements are used to describe a database, to define its structure, to create its objects, and to create the subobjects of the table. You can make changes to a rule set after you create it.

Today's database industry embeds DDL in any formal language that describes data. However, it is considered a subset of SQL (Structured Query Language). SQL often uses normal English imperative verbs as sentences to implement modifications to the database. Therefore, DDL does not appear as a different language in an SQL database, but it does define changes to the database schema.

2. Practice objective

Review the notion of Oracle DDL statements creating other types of object.

3. Developing

Activity 1: Read all the choices carefully because there might be more than one correct answer. Choose all the correct answers for each question. Explain the reason for your answer.

CREATE PRIVATE AND PUBLIC SYNONYMS

1. What are distinguishing characteristics of a public synonym rather than a private synonym? Challenge question.

R = B) Public synonyms can be accessed by name without a schema name qualifier.

D) Public synonyms can have the same names as tables or views.

Public Synonyms: Synonyms that can be accessed by all users of the database. There is no doubt that only plan users with DBA authority can set them.

Private Synonyms: synonyms that can only be accessed by the user who created them.

2. Consider these three statements:

create synonym s1 for staff;

create public synonym s1 for warehouse;

*select * from s1;*

Which of the following statements is correct? (Choose the best answer.)

R = C) The third statement will show the contents of staff.

A) The second statement will fail because an object S1 already exists.

Synonyms with the same name cannot be created for different objects.

3. A view and a synonym are created as follows:

*create view dept_v as select * from dept;*

create synonym dept_s for dept_v;

Subsequently the table DEPT is dropped. What will happen if you query the synonym DEPT_S? (Choose the best answer.)

R = D) There will be an error because the view will be invalid.

CREATE, MAINTAIN, AND USE SEQUENCES

4. A sequence is created as follows:

create sequence seq1 maxvalue 100;

If the current value is already 100, when you attempt to select SEQ1.NEXTVAL what will happen? (Choose the best answer.)

R = D) There will be an error.

There will be an error since the sequence will have reached the maximum value and because by default the sequences are not cycled.

5. You create a sequence as follows:

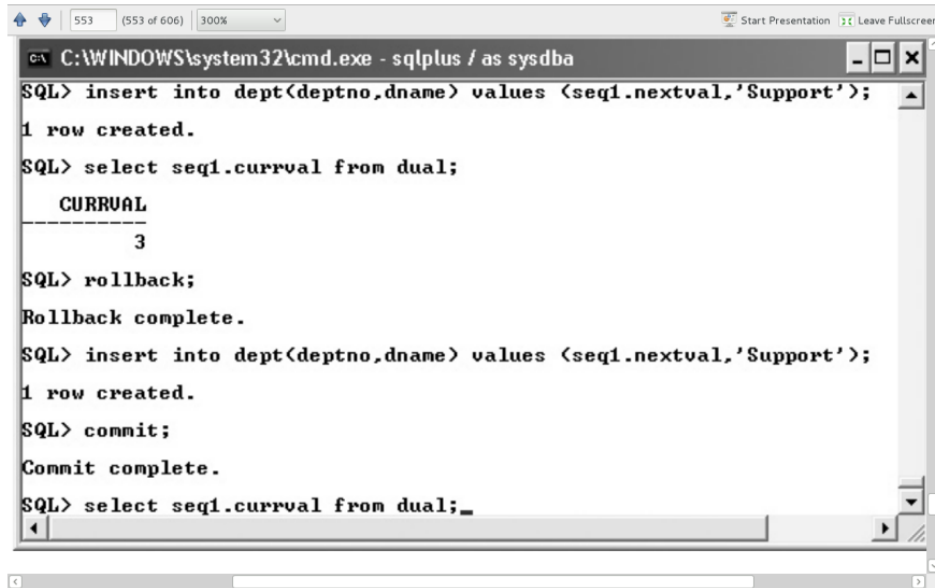
create sequence seq1 start with 5;

After selecting from it a few times, you want to reinitialize it to reissue the numbers already generated. How can you do this? (Choose the best answer.)

R = A) You must drop and re-create the sequence.

In this case, the sequence should be deleted and recreated, but now it could be cycled.

6. Study the following exhibit:



```
C:\WINDOWS\system32\cmd.exe - sqlplus / as sysdba
SQL> insert into dept(deptno,dname) values (seq1.nextval,'Support');
1 row created.
SQL> select seq1.currval from dual;
  CURRVAL
-----
        3
SQL> rollback;
Rollback complete.
SQL> insert into dept(deptno,dname) values (seq1.nextval,'Support');
1 row created.
SQL> commit;
Commit complete.
SQL> select seq1.currval from dual;
```

Figura 1: Image for question six

Assuming that the sequence SEQ1 was created with the option ORDER and INCREMENT BY set to 1, what value will be returned by the final SELECT statement? (Choose the best answer.)

R = B) 3

This is because the ROLLBACK statement allows you to undo all the modifications that have been made to the Database but have not been written to the Hard Disk by the COMMIT statement, that is, it removes from memory all the changes made in the Database until the last COMMIT that was made.

CREATE AND MAINTAIN INDEXES

7. A UNIQUE constraint on a column requires an index. Which of the following scenarios is correct? (Choose one or more correct answers.)

R = A) If a UNIQUE index already exists on the column, it will be used.

D) If any index exists on the column, there will be an error as Oracle attempts to create another index implicitly.

If a unique index already exists on a column, it is no longer possible to create another unique index for that column.

8. This statement will fail:

create unique bitmap index on employees(department_id,hire_date);

R = A) Bitmap indexes cannot be unique.

A main feature of BITMAP indexes is that they cannot be unique.

9. You have created an index with this statement:

create index ename_i on employees(last_name,first_name);

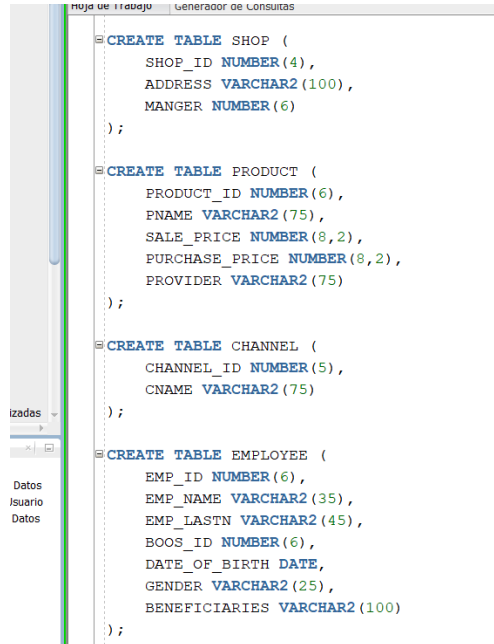
How can you adjust the index to include the employees' birthdays, which is a date type column called DOB? (Choose the best answer.)

R = C) You must drop the index and re-create it

I chose this option since there is no way to add another column to the index since the statement of option A is not correct.

Activity 2:

Figures 2 and 3 show the DDL statements for creating the tables.



```

CREATE TABLE SHOP (
  SHOP_ID NUMBER(4),
  ADDRESS VARCHAR2(100),
  MANGER NUMBER(6)
);

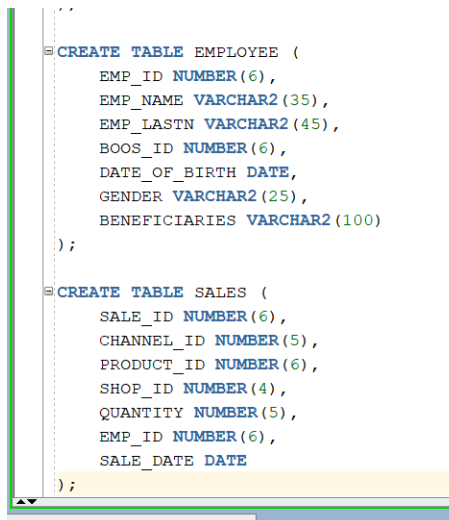
CREATE TABLE PRODUCT (
  PRODUCT_ID NUMBER(6),
  PNAME VARCHAR2(75),
  SALE_PRICE NUMBER(8,2),
  PURCHASE_PRICE NUMBER(8,2),
  PROVIDER VARCHAR2(75)
);

CREATE TABLE CHANNEL (
  CHANNEL_ID NUMBER(5),
  CNAME VARCHAR2(75)
);

CREATE TABLE EMPLOYEE (
  EMP_ID NUMBER(6),
  EMP_NAME VARCHAR2(35),
  EMP_LASTN VARCHAR2(45),
  BOOS_ID NUMBER(6),
  DATE_OF_BIRTH DATE,
  GENDER VARCHAR2(25),
  BENEFICIARIES VARCHAR2(100)
);

```

Figura 2: DDL statements for creating the tables.



```

CREATE TABLE EMPLOYEE (
  EMP_ID NUMBER(6),
  EMP_NAME VARCHAR2(35),
  EMP_LASTN VARCHAR2(45),
  BOOS_ID NUMBER(6),
  DATE_OF_BIRTH DATE,
  GENDER VARCHAR2(25),
  BENEFICIARIES VARCHAR2(100)
);

CREATE TABLE SALES (
  SALE_ID NUMBER(6),
  CHANNEL_ID NUMBER(5),
  PRODUCT_ID NUMBER(6),
  SHOP_ID NUMBER(4),
  QUANTITY NUMBER(5),
  EMP_ID NUMBER(6),
  SALE_DATE DATE
);

```

Figura 3: DDL statements for creating the tables.

Figure 4 shows the DDL statements for the creation of the indexes of the columns of the tables.

```
-- INDICES PARA LAS COLUMNAS DE LAS TABLAS
CREATE INDEX SHOP_ID_IDX ON SHOP (SHOP_ID);
CREATE BITMAP INDEX SHOP_MANAGER_IDX ON SHOP (MANGER);
CREATE BITMAP INDEX PRO_ID_IDX ON PRODUCT (PRODUCT_ID);
CREATE INDEX PRO_NAME_IDX ON PRODUCT (PNAME);
CREATE INDEX PRO_PRICE_IDX ON PRODUCT (SALE_PRICE, PURCHASE_PRICE);
CREATE INDEX CHANNEL_ID_IDX ON CHANNEL (CHANNEL_ID);
CREATE BITMAP INDEX EMP_ID_IDX ON EMPLOYEE (EMPLOYEE_ID);
CREATE INDEX EMP_NAME_IDX ON EMPLOYEE (EMP_NAME, EMP_LASTN);
CREATE BITMAP INDEX EMP_BOOD_ID_IDX ON EMPLOYEE (BOOS_ID);
CREATE BITMAP INDEX SALES_SALE_ID_IDX ON SALES (SALE_ID);
CREATE INDEX SAL_CHANNEL_ID_IDX ON SALES (CHANNEL_ID);
CREATE BITMAP INDEX SALES_PRO_ID_IDX ON SALES (PRODUCT_ID);
CREATE INDEX SALES_SHOP_ID_IDX ON SALES (SHOP_ID);
CREATE BITMAP INDEX SALES_EMP_ID_IDX ON SALES (EMP_ID);
CREATE BITMAP INDEX SALES_SALE_DATE_IDX ON SALES (SALE_DATE);
```

Figura 4: *DDL statements for creating indexes.*

In the mangaer, product.id, employee.id, boss.id, sale.id and sake_date columns, BitMap indexes were created since these columns have a low cardinality and can also offer significant savings in terms of space, as well as very good query performance.

Figure 5 shows the DDL statements for creating the primary keys of the tables.

```
-- LLAVES PRIMARIAS
ALTER TABLE SHOP ADD CONSTRAINT SHOP_PK PRIMARY KEY (SHOP_ID);
ALTER TABLE PRODUCT ADD CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID);
ALTER TABLE CHANNEL ADD CONSTRAINT CHANNEL_PK PRIMARY KEY (CHANNEL_ID);
ALTER TABLE EMPLOYEE ADD CONSTRAINT EMPLOYEE_PK PRIMARY KEY (EMP_ID);
ALTER TABLE SALES ADD CONSTRAINT SALES_PK PRIMARY KEY (SALE_ID);
```

Figura 5: *DDL statements, primary keys.*

Figure 6 shows the DDL statements for creating the foreign keys of the tables.

```
-- LLAVES FORANEAS
ALTER TABLE SHOP ADD CONSTRAINT SHOP_MANAGER_FK FOREIGN KEY (MANGER)
REFERENCES EMPLOYEE (EMP_ID);

ALTER TABLE EMPLOYEE ADD CONSTRAINT BOSS_EMP_FK FOREIGN KEY (BOOS_ID)
REFERENCES EMPLOYEE (EMP_ID);

ALTER TABLE SALES ADD CONSTRAINT CHANNEL_SALES_FK FOREIGN KEY (CHANNEL_ID)
REFERENCES CHANNEL (CHANNEL_ID);

ALTER TABLE SALES ADD CONSTRAINT PRODUCT_SALES_FK FOREIGN KEY (PRODUCT_ID)
REFERENCES PRODUCT (PRODUCT_ID);

ALTER TABLE SALES ADD CONSTRAINT SHOP_SALES_FK FOREIGN KEY (SHOP_ID)
REFERENCES SHOP (SHOP_ID);

ALTER TABLE SALES ADD CONSTRAINT EMP_SALES_FK FOREIGN KEY (EMP_ID)
REFERENCES EMPLOYEE (EMP_ID);
```

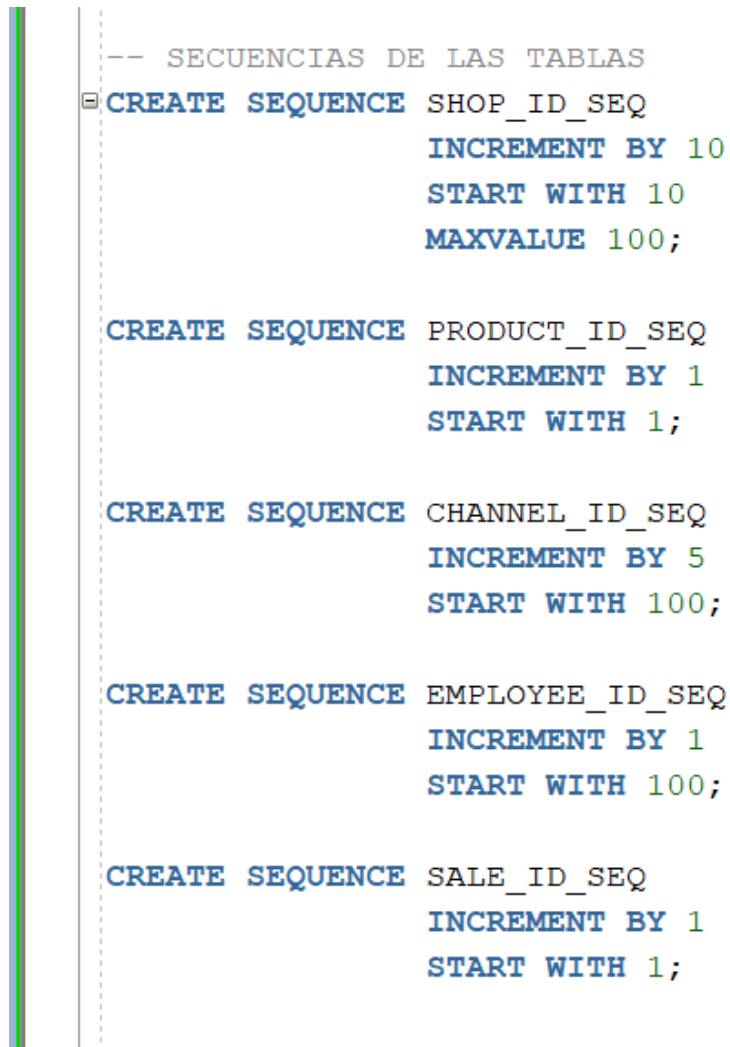
Figura 6: *DDL statements, foreign keys.*

Figure 7 shows the DDL statements for the creation of the not null restrictions of the different columns of the tables.

```
-- COLUMNAS NOT NULL
ALTER TABLE SHOP MODIFY ADDRESS NOT NULL;
ALTER TABLE SHOP MODIFY MANGER NOT NULL;
ALTER TABLE PRODUCT MODIFY PNAME NOT NULL;
ALTER TABLE PRODUCT MODIFY SALE_PRICE NOT NULL;
ALTER TABLE PRODUCT MODIFY PROVIDER;
ALTER TABLE CHANNEL MODIFY CNAME NOT NULL;
ALTER TABLE EMPLOYEE MODIFY EMP_NAME NOT NULL;
ALTER TABLE EMPLOYEE MODIFY EMP_LASTN NOT NULL;
ALTER TABLE EMPLOYEE MODIFY DATE_OF_BIRTH NOT NULL;
ALTER TABLE EMPLOYEE MODIFY GENDER NOT NULL;
ALTER TABLE EMPLOYEE MODIFY BENEFICIARIES NOT NULL;
ALTER TABLE SALES MODIFY QUANTITY NOT NULL;
ALTER TABLE SALES MODIFY SALE_DATE NOT NULL;
```

Figura 7: *DDL statements, not null constraints.*

Figure 8 shows the DDL statements for creating the sequences for the primary keys of the tables.

The image shows a screenshot of a SQL development environment. On the left, there is a vertical toolbar with a blue and green bar. The main area displays SQL code for creating sequences. The code is as follows:

```
-- SECUENCIAS DE LAS TABLAS
CREATE SEQUENCE SHOP_ID_SEQ
    INCREMENT BY 10
    START WITH 10
    MAXVALUE 100;

CREATE SEQUENCE PRODUCT_ID_SEQ
    INCREMENT BY 1
    START WITH 1;

CREATE SEQUENCE CHANNEL_ID_SEQ
    INCREMENT BY 5
    START WITH 100;

CREATE SEQUENCE EMPLOYEE_ID_SEQ
    INCREMENT BY 1
    START WITH 100;

CREATE SEQUENCE SALE_ID_SEQ
    INCREMENT BY 1
    START WITH 1;
```

Figura 8: *DDL statements, sequences of the tables.*

Figure 9 shows the DDL statements for creating the synonyms for each of the tables.

```
-- SINONIMOS PARA LAS TABLAS
CREATE SYNONYM SH FOR SHOP;
CREATE SYNONYM P FOR PRODUCT;
CREATE SYNONYM CH FOR CHANNEL;
CREATE SYNONYM E FOR EMPLOYEE;
CREATE SYNONYM SA FOR SALES;
```

Figura 9: *DDL statements, sequences of the tables.*

Activity 3:

Figure 10 shows the DDL statements for creating the indexes. The indices are of type B-tree since it is assumed that the indices are going to be used for fast searches and not for storage improvement.

```
-- INDICES
CREATE INDEX VEH_ID_IDX ON VEHICLE (ID_VEH);
CREATE INDEX VEH_LIC_PLATES_IDX ON VEHICLE (LIC_PLATES);
CREATE INDEX VEH_PLATES_OWNER_ID_IDX ON VEHICLE (LIC_PLATES, OWNER_ID);
CREATE INDEX OWNER_ID_IDX ON OWNER (OWNER_ID);
CREATE INDEX OWNER_FORENAME_SURNAME_IDX ON OWNER (FORENAME, SURNAME);
```

Figura 10: *DDL statements for creating indexes.*

Figure 11 shows the DDL statements for the creation of the different constraints of the tables which are (primary keys, foreign keys and constraint not null).

```
-- PRIMARY KEYS
ALTER TABLE VEHICLE ADD CONSTRAINT VEHICLE_PK PRIMARY KEY (ID_VEH);
ALTER TABLE OWNER ADD CONSTRAINT OWNER_PK PRIMARY KEY (OWNER_ID);

-- CONSTRAINT NOT NULL
ALTER TABLE VEHICLE MODIFY LIC_PLATES NOT NULL;
ALTER TABLE VEHICLE MODIFY OWNER_ID NOT NULL;
ALTER TABLE OWNER MODIFY SURNAME NOT NULL;
ALTER TABLE OWNER MODIFY FORENAME NOT NULL;
ALTER TABLE OWNER MODIFY DATEOBIRTH NOT NULL;

-- FOREIGN KEYS
ALTER TABLE VEHICLE ADD CONSTRAINT OWNER_VEHICLE_FK FOREIGN KEY (OWNER_ID)
REFERENCES OWNER (OWNER_ID);
```

Figura 11: *DDL statements for constraints.*

Activity 4:

Propose a response to the following issues:

- You are involved in designing a database to be used for online order entry and offline financial reporting. What should you consider with regard to data consulting, synonyms, and indexes?
- Should sequences always be used for primary keys?

Responding to the first question and regarding the indices, I would propose the use of B-tree type indices for searches (queries) of clients (full name), employees, orders, etc. Although Bitmap type indexes could also be used since, as we are going to have hundreds of orders and customers, good storage management will be needed, another point to take into account would be the cardinality of each of our columns in each table in order to know what type of indices to use.

In this case, the use of synonyms is also recommended as this helps to simplify the writing of DDL and DML statements.

Regarding the second question about whether sequences should always be used for primary keys, in my opinion yes, since this guarantees that the data does not have inconsistencies as long as the primary keys are numeric and the sequences are created in a right way.

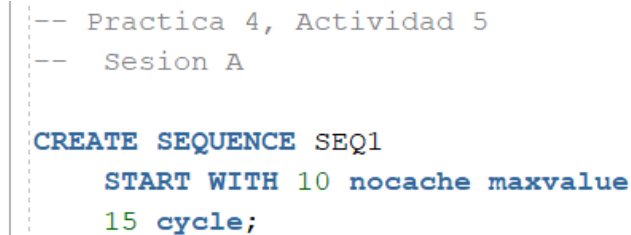
Activity 5:

In this exercise, you will create some sequences and use them. You will need two concurrent sessions.

2. In your A session, create a sequence as follows:

create sequence seq1 start with 10 nocache maxvalue 15 cycle;

See figure 12

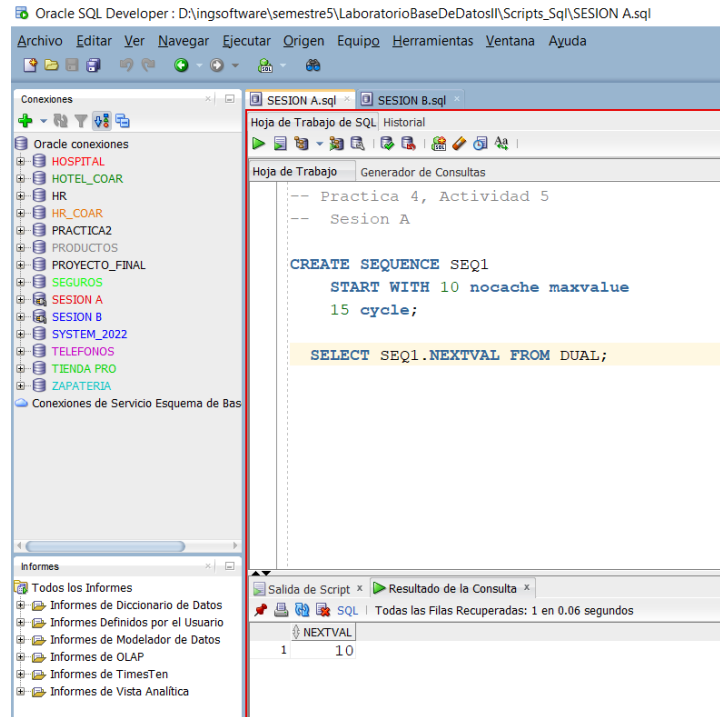
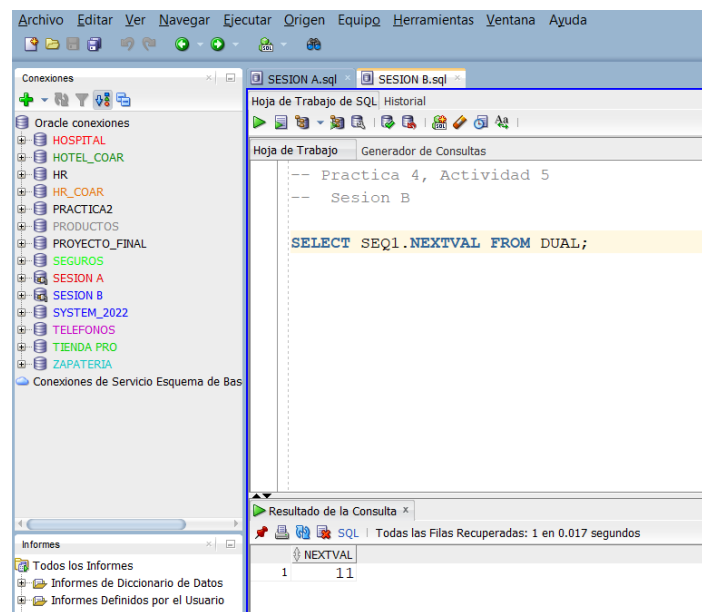
The image shows a screenshot of a terminal window with a light gray background. On the left side, there is a vertical dashed line. The text in the terminal is as follows: two comment lines starting with '--' (Practica 4, Actividad 5 and Sesion A), followed by a SQL statement to create a sequence. The SQL statement is: 'CREATE SEQUENCE SEQ1' on one line, 'START WITH 10 nocache maxvalue' on the next, and '15 cycle;' on the third. The SQL keywords are in blue, and the numbers and punctuation are in green. The bottom of the terminal window has a yellow highlight bar.

```
-- Practica 4, Actividad 5
-- Sesion A

CREATE SEQUENCE SEQ1
  START WITH 10 nocache maxvalue
  15 cycle;
```

Figura 12: *DDL statements for constraints.*

3. Use of the sequence: Observe the following figures

Figura 13: *Use of the sequence.*Figura 14: *Use of the sequence.*

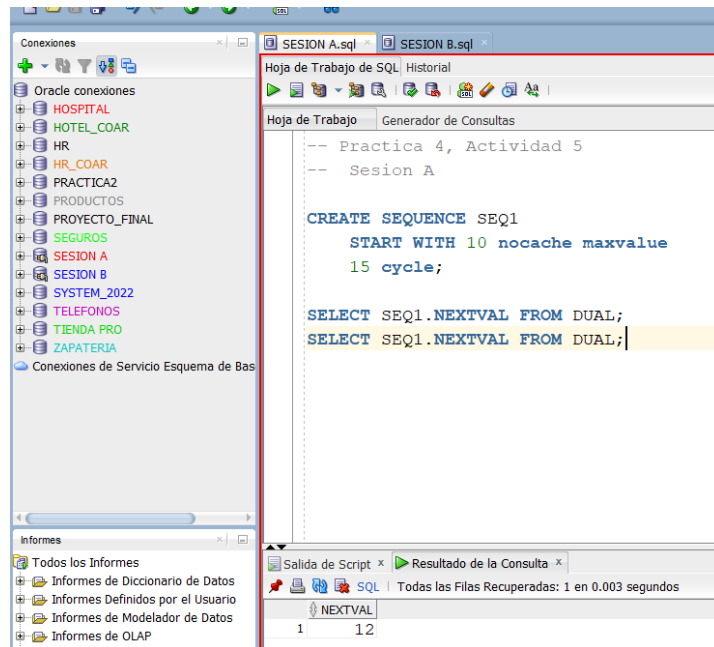


Figura 15: Use of the sequence.

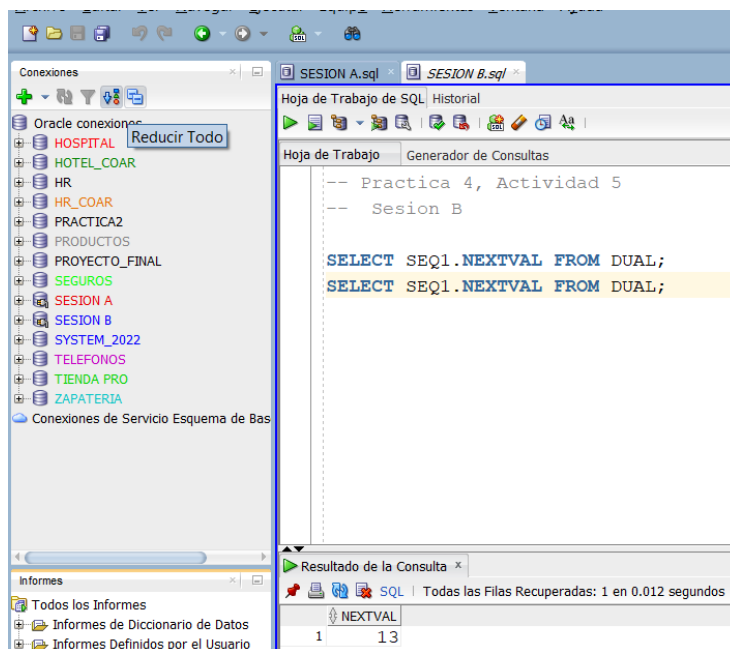


Figura 16: Use of the sequence.

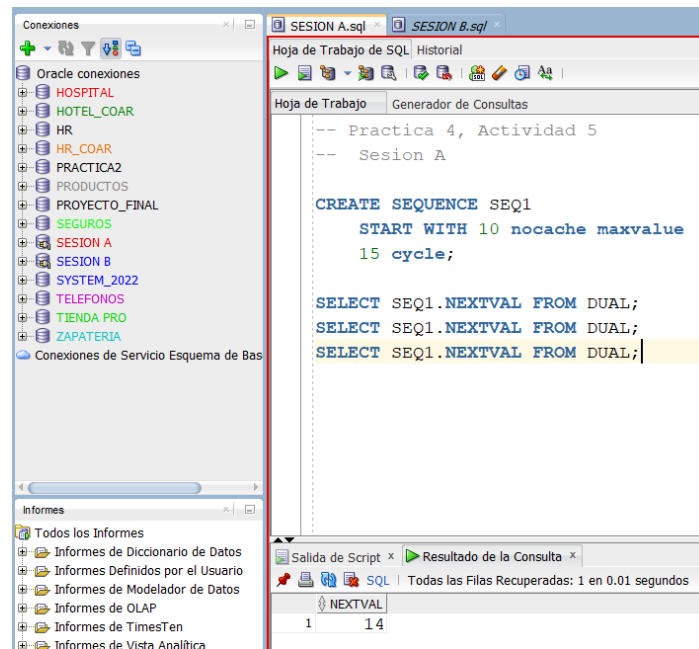


Figura 17: Use of the sequence.

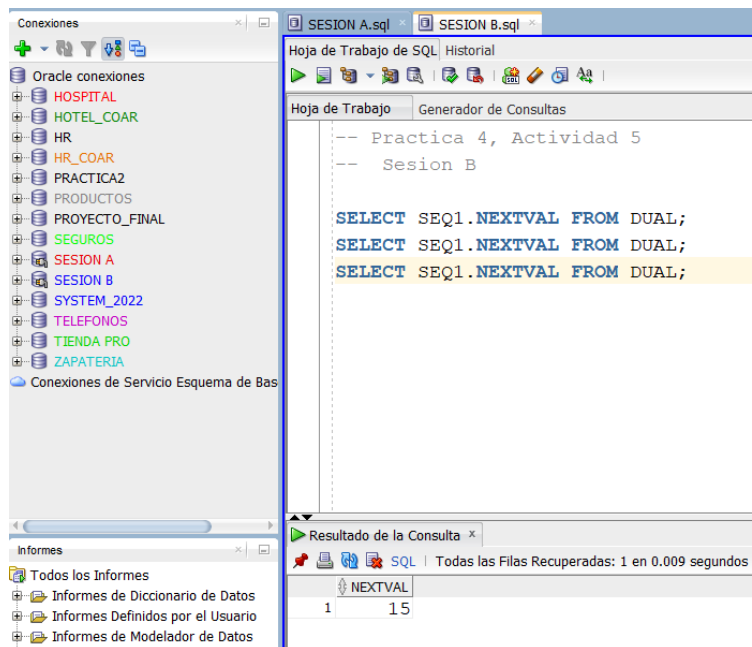


Figura 18: Use of the sequence.

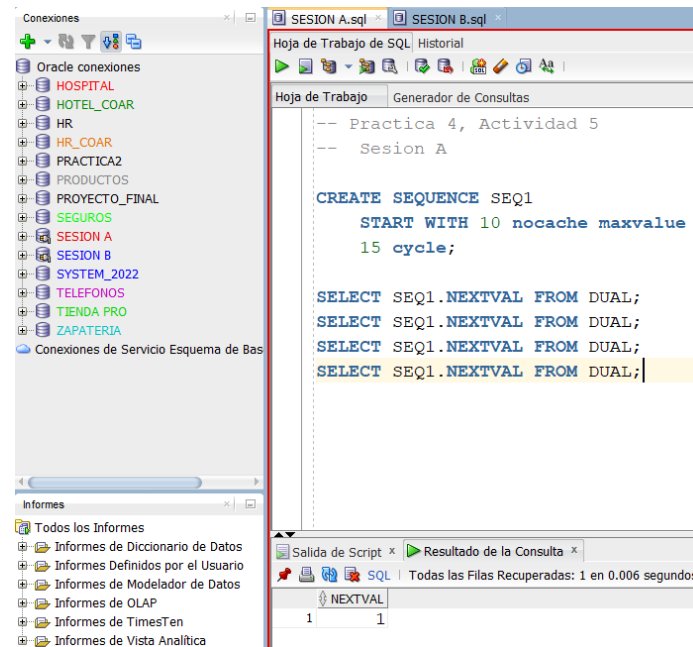


Figura 19: Use of the sequence.

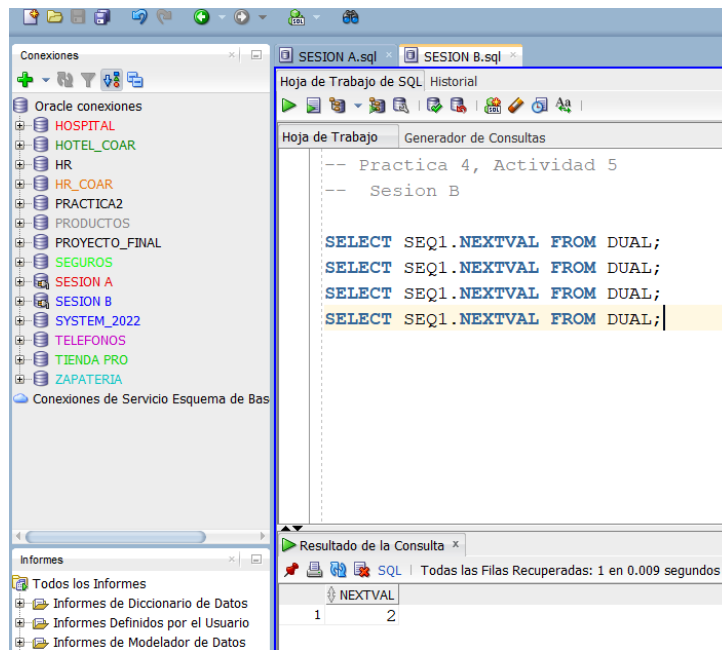


Figura 20: Use of the sequence.

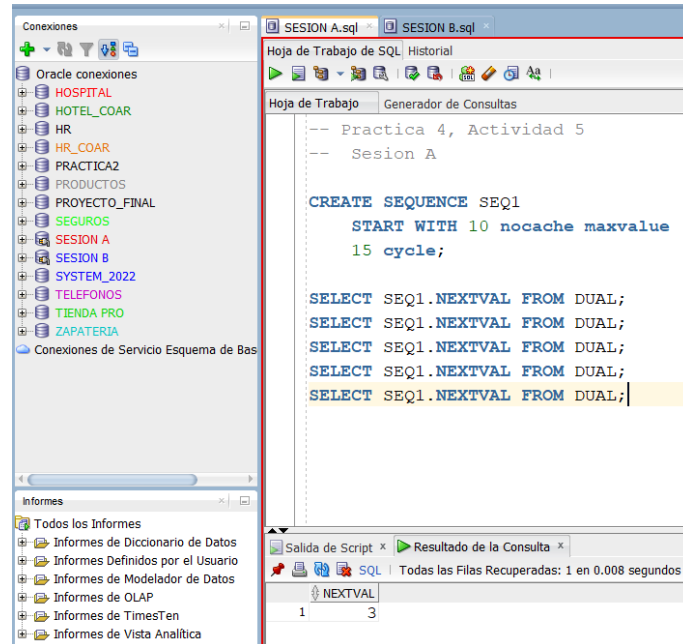


Figura 21: Use of the sequence.

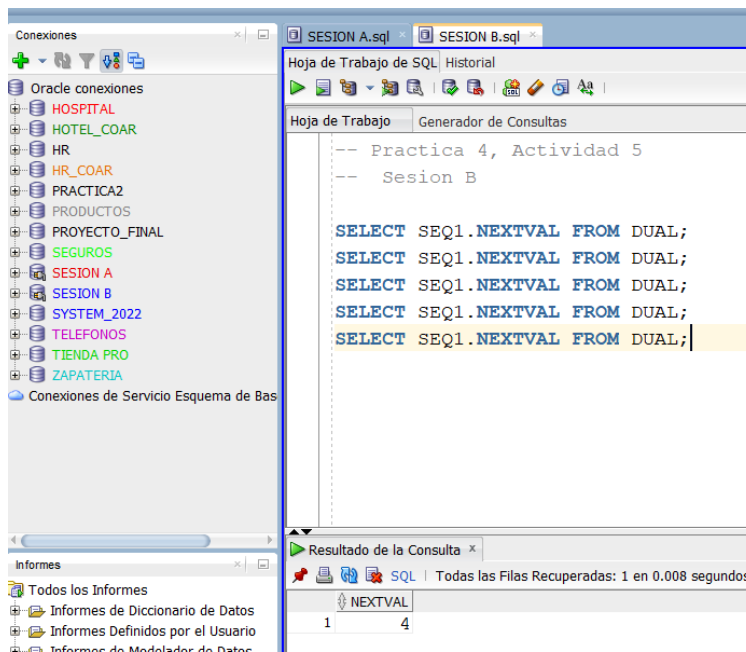


Figura 22: Use of the sequence.

4. Create a table with a primary key:

```
create table seqtest(c1 number,c2 varchar2(10));  
alter table seqtest add constraint seqtest_pk  
primary key (c1);
```

Figura 23: *Creating a table.*

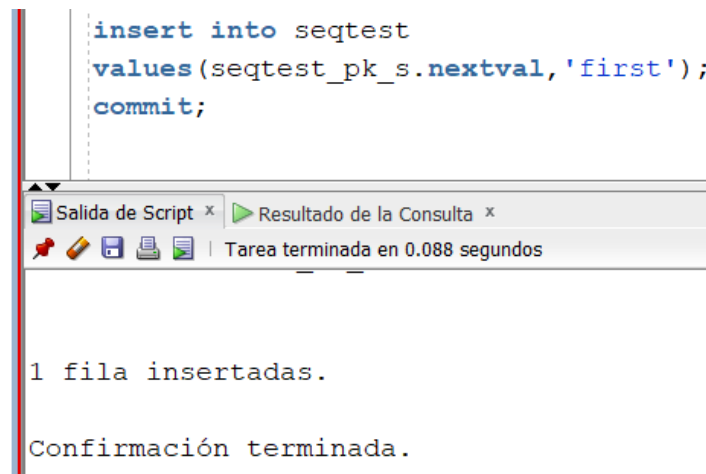
5. Create a sequence to generate primary key values:

```
create sequence seqtest_pk_s;
```

Figura 24: *Create a sequence.*

6. In your A session, insert a row into the new table and commit:

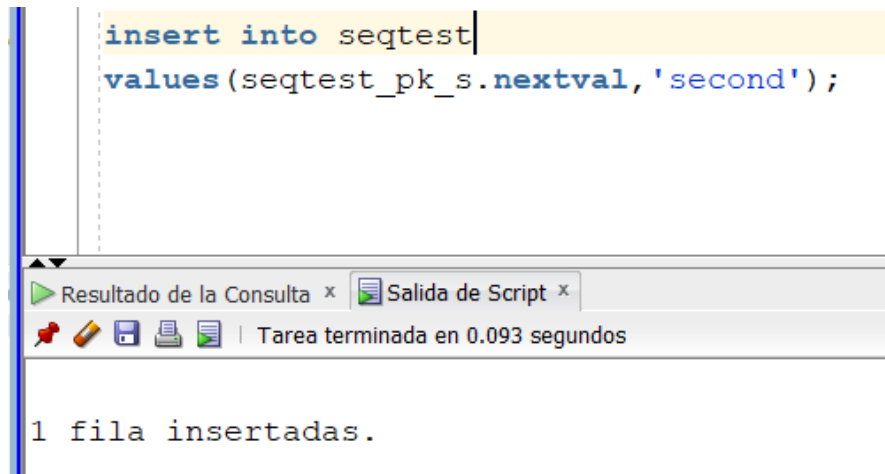
```
insert into seqtest  
values(seqtest_pk_s.nextval,'first');  
commit;
```



1 fila insertadas.
Confirmación terminada.

Figura 25: *Inserting data.*

7. In your B session, insert a row into the new table and do not commit it:



```
insert into seqtest
values(seqtest_pk_s.nextval, 'second');
```

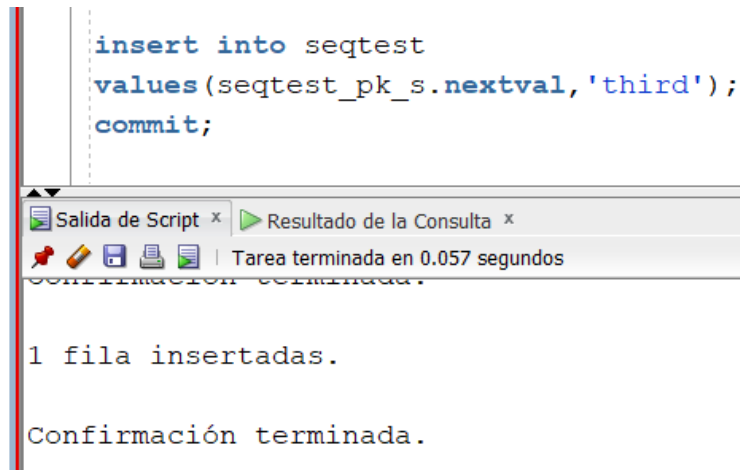
Resultado de la Consulta x Salida de Script x

Tarea terminada en 0.093 segundos

1 fila insertadas.

Figura 26: *Inserting data, no confirmation.*

8. In your A session, insert a third row and commit:



```
insert into seqtest
values(seqtest_pk_s.nextval, 'third');
commit;
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0.057 segundos

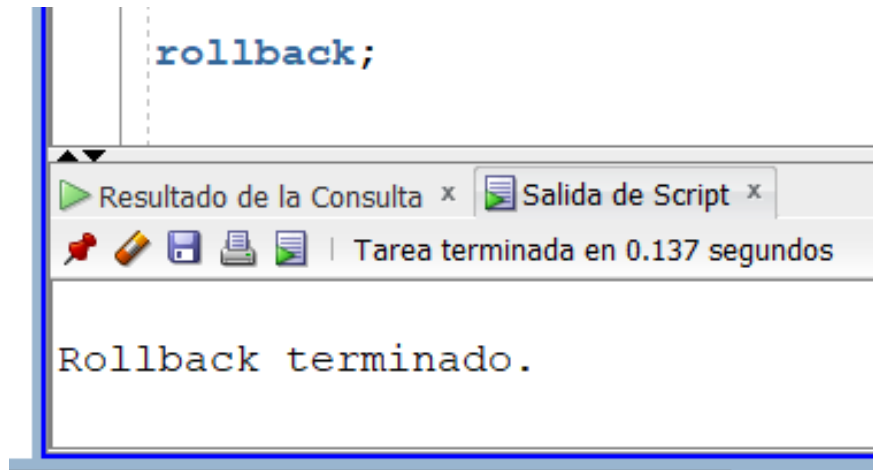
Confirmación terminada.

1 fila insertadas.

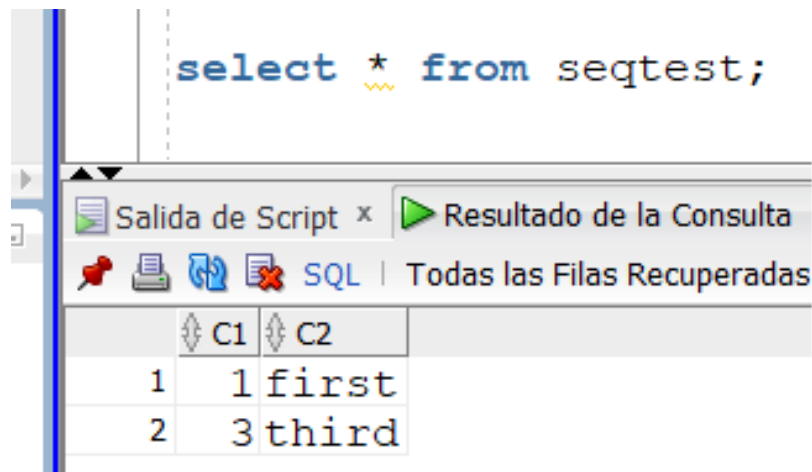
Confirmación terminada.

Figura 27: *Inserting data.*

9. In your B session, roll back the second insertion:

Figura 28: *Rollback.*

10. In your B session, see the contents of the table:

Figura 29: *Querying data.*

This demonstrates that sequences are incremented and the next value published immediately, outside the transaction control mechanism.

11. Tidy up:

```
drop table seqtest;  
drop sequence seqtest_pk_s;  
drop sequence seq1;
```

Figura 30: *Removed.*

Activity 6:

Execute the following sentences and include an image for each one:

Creating Indexes

In this exercise, create indexes on a copy of the EMPLOYEES table in the HR schema.

1. Connect to your database as your user.
2. Create a table that is a copy of HR.EMPLOYEES:

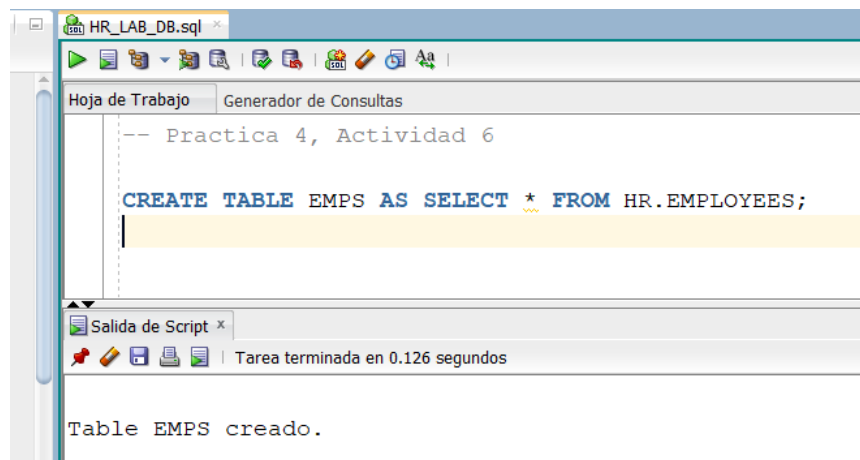
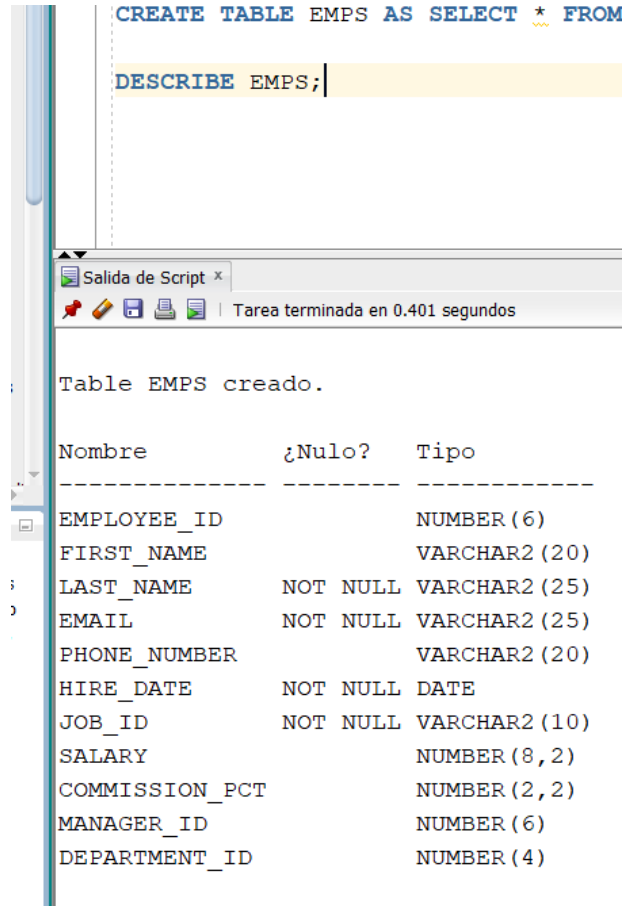


Figura 31: *Copying the employees table.*

This table will have neither indexes nor primary, unique, or foreign key constraints, because these

are not copied by a CREATE TABLE AS command. The NOT NULL constraints will have been copied. Confirm this by describing the table:

describe emps;



```
CREATE TABLE EMPS AS SELECT * FROM  
DESCRIBE EMPS;
```

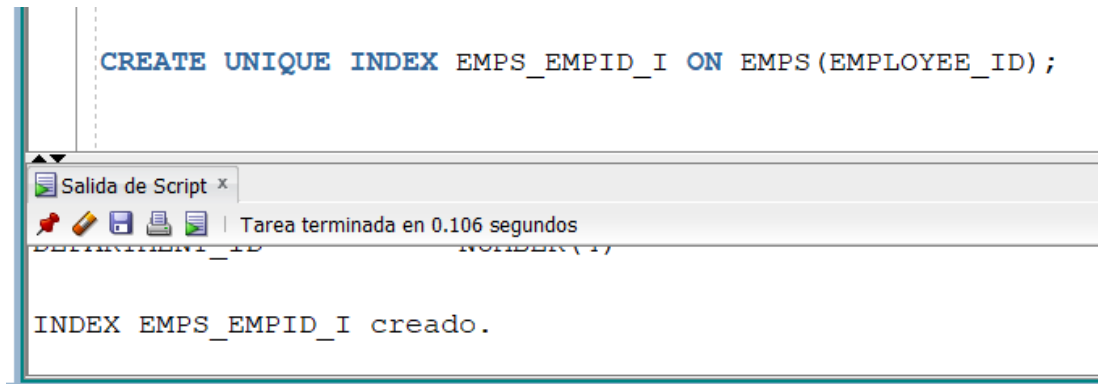
Salida de Script x
Tarea terminada en 0.401 segundos

Table EMPS creado.

Nombre	¿Nulo?	Tipo
EMPLOYEE_ID		NUMBER (6)
FIRST_NAME		VARCHAR2 (20)
LAST_NAME	NOT NULL	VARCHAR2 (25)
EMAIL	NOT NULL	VARCHAR2 (25)
PHONE_NUMBER		VARCHAR2 (20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2 (10)
SALARY		NUMBER (8,2)
COMMISSION_PCT		NUMBER (2,2)
MANAGER_ID		NUMBER (6)
DEPARTMENT_ID		NUMBER (4)

Figura 32: *Description of the table emps.*

3. Create an index to be used for the primary key constraint:



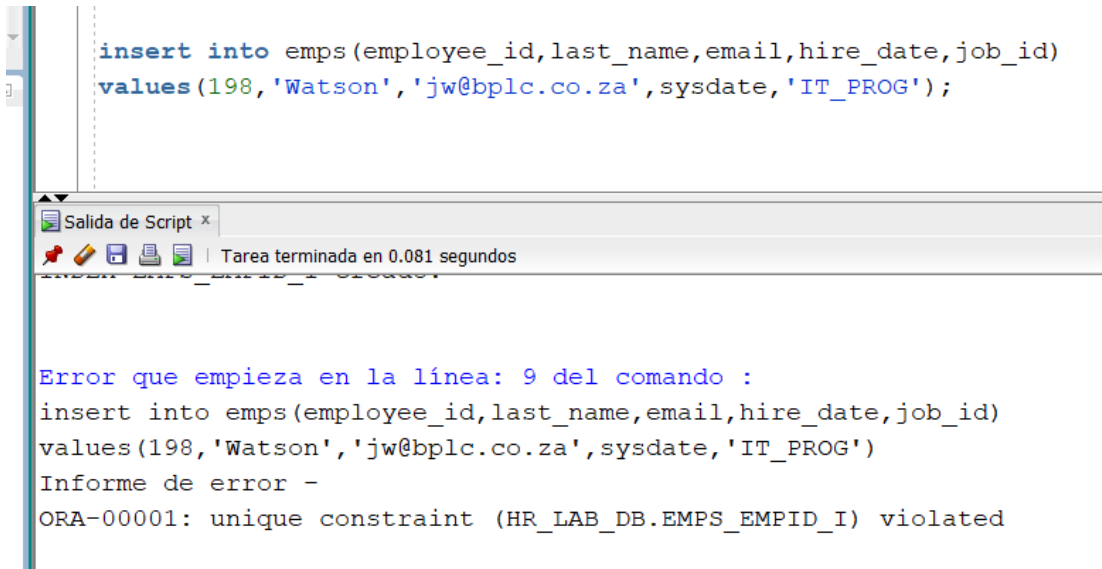
```
CREATE UNIQUE INDEX EMPS_EMPID_I ON EMPS(EMPLOYEE_ID);
```

Salida de Script x
Tarea terminada en 0.106 segundos

```
INDEX EMPS_EMPID_I creado.
```

Figura 33: *Creating an index.*

4. Demonstrate that a unique index cannot accept duplicates, even before a constraint is defined:



```
insert into emps(employee_id,last_name,email,hire_date,job_id)
values(198,'Watson','jw@bplc.co.za',sysdate,'IT_PROG');
```

Salida de Script x
Tarea terminada en 0.081 segundos

```
Error que empieza en la línea: 9 del comando :
insert into emps(employee_id,last_name,email,hire_date,job_id)
values(198,'Watson','jw@bplc.co.za',sysdate,'IT_PROG')
Informe de error -
ORA-00001: unique constraint (HR_LAB_DB.EMPS_EMPID_I) violated
```

Figura 34: *Inserting data.*

This will return an error because the index cannot insert a second employee_id 198. Index uniqueness is an attribute of the index that can exist without a constraint but should not be relied upon to enforce data integrity

5. Create additional indexes on columns that are likely to be used in WHERE clauses,

using B*Tree for columns of high cardinality and bitmap for columns of low cardinality:

```
create index emps_name_i on emps(last_name,first_name);  
create index emps_tel_i on emps(phone_number);  
create bitmap index emps_mgr_i on emps(manage_id);  
create bitmap index emps_dept_i on emps(department_id);
```

Figura 35: *Creating indexes.*

6. Define some constraints:

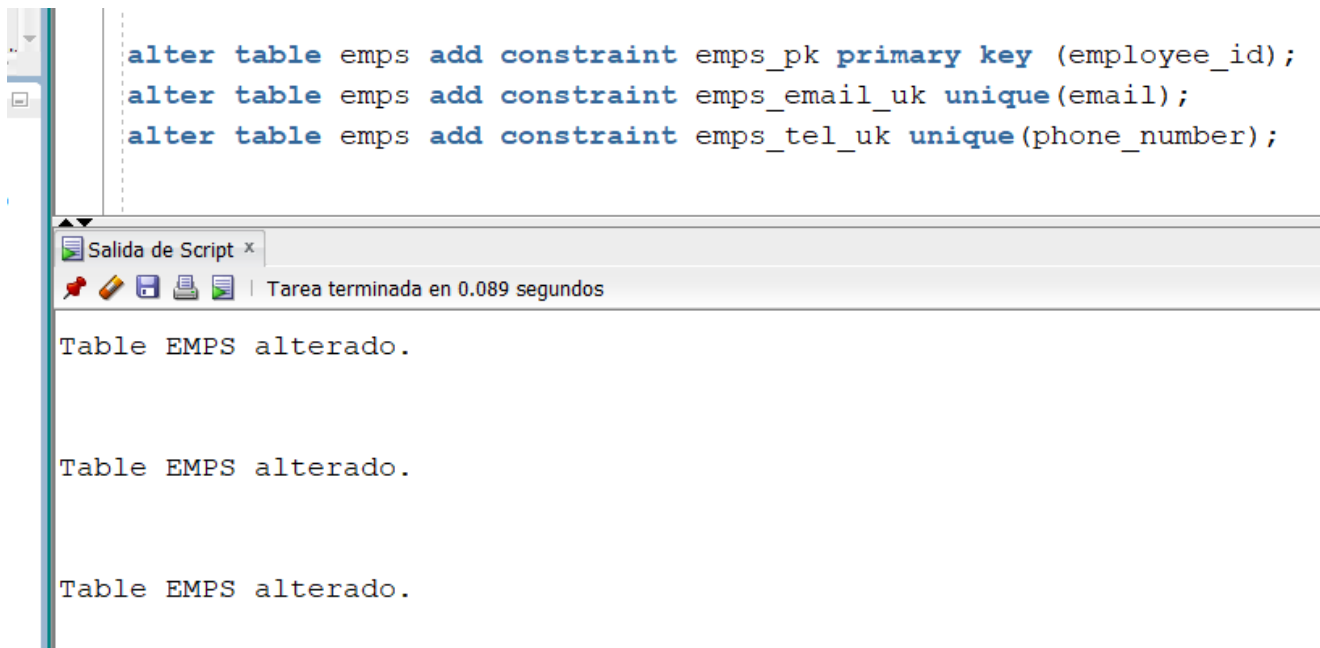
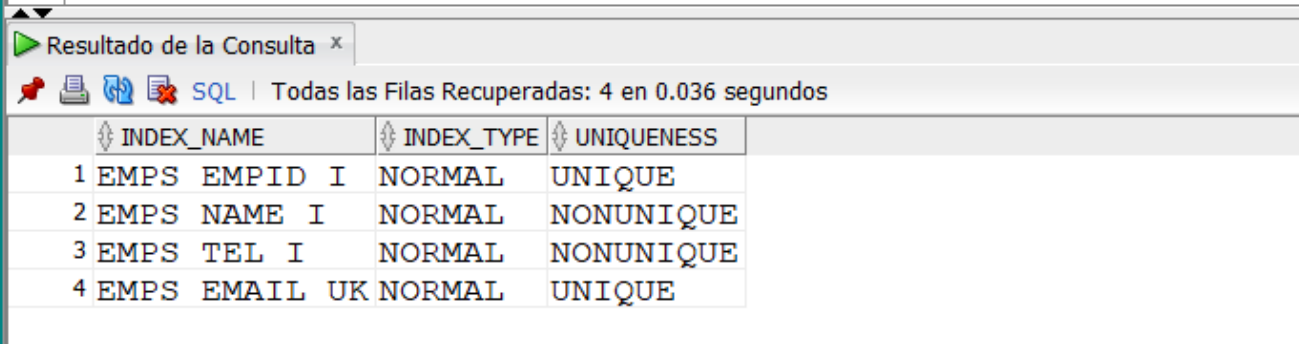


Figura 36: *Creating Constraints.*

7. Display the index names and their type:

```
select index_name,index_type,uniqueness from user_indexes  
where table_name='EMPS';
```



Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 4 en 0.036 segundos

	INDEX_NAME	INDEX_TYPE	UNIQUENESS
1	EMPS EMPID I	NORMAL	UNIQUE
2	EMPS NAME I	NORMAL	NONUNIQUE
3	EMPS TEL I	NORMAL	NONUNIQUE
4	EMPS EMAIL UK	NORMAL	UNIQUE

Figura 37: *Types of indexes.*

The view USER_INDEXES shows details of all indexes in your current schema. Note that in addition to the five indexes explicitly created in steps 3 and 5, there is also an index created implicitly with the name of the constraint defined on EMAIL. Note also that the unique constraint on PHONE_NUMBER is being enforced with a nonunique index; this is perfectly possible, because although the constraint mechanism uses indexes, it is independent of the structure of the index.

8. Tidy up by dropping the EMPS table, and confirm that all the indexes have also gone:

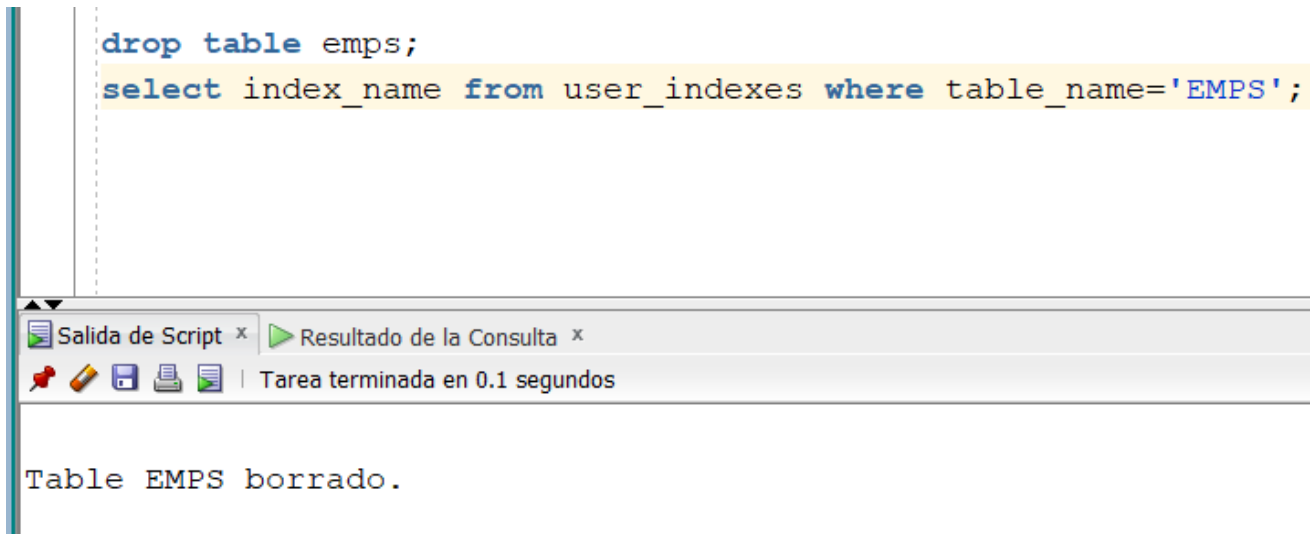


Figura 38: Deleted emps table.

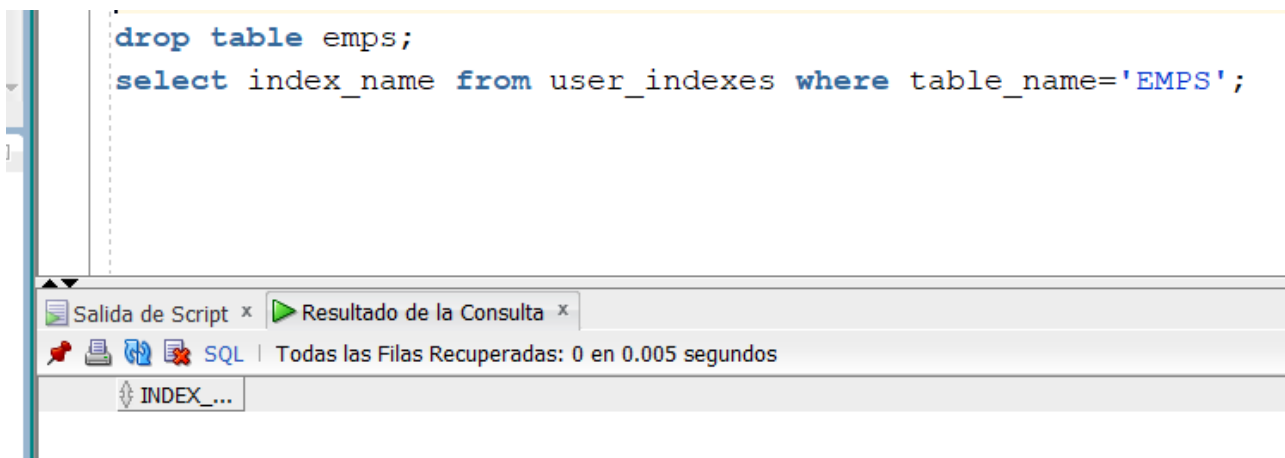


Figura 39: Showing the indices and their type.

4. Pre-assessment

In this section you will find the Pre-assessment

Criteria to be evaluate	Does it comply?	(%)
COMPLIES WITH THE REQUESTED FUNCTIONALITY	YES	
HAS THE CORRECT INDENTATION	YES	
HAS AN EASY WAY TO ACCESS THE PROVIDED FILES	YES	
HAS A REPORT WITH IDC FORMAT	YES	
REPORT INFORMATION IS FREE OF SPELLING ERRORS	YES	
DELIVERED IN TIME AND FORM	YES	
IS FULLY COMPLETED (SPECIFY THE PERCENTAGE COMPLETED)	YES	95 %

5. Conclusion

The Oracle DDL language is transcendental in the handling of SQL statements at the level of both administrator and database programmer, since it allows the definition of database schemes regardless of the platform used to generate it.

This practice was very important for me since it helped me to remember and reinforce my knowledge in the use of DDL statements.