

Bases de Datos 1

Laboratorio 4

- Fabián Bustos
- Ian Murillo

Evidencia Laboratorio 4

Ejercicio 1:

Cree un esquema nuevo llamado UN con sus correspondientes tablespaces (índices y datos).

Accediendo desde system a los tablespaces de la base de datos se pueden observar los tablespaces UN_IND y UN_DATA:

1

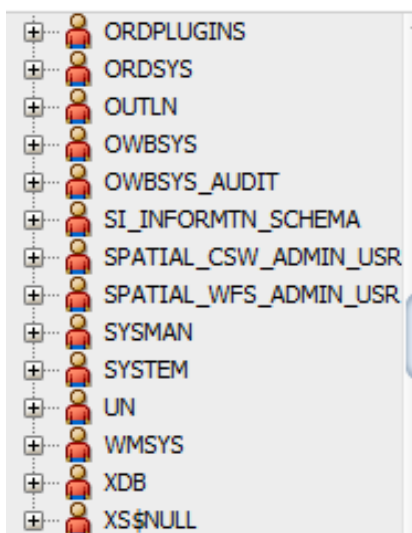
```
select * from dba_tablespaces
```

Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 9 en 0,088 segundos

	TABLESPACE_NAME	BLOCK_SIZE	INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENTS	MAX_EXTENTS	MAX_SIZE	PCT_INCREASE	MIN_EXTLEN	STATU
1	SYSTEM	8192	65536	(null)	1	2147483645	2147483645	(null)	65536	ONLINE
2	SYSAUX	8192	65536	(null)	1	2147483645	2147483645	(null)	65536	ONLINE
3	UNDOTBS1	8192	65536	(null)	1	2147483645	2147483645	(null)	65536	ONLINE
4	TEMP	8192	1048576	1048576	1	(null)	2147483645	0	1048576	ONLINE
5	USERS	8192	65536	(null)	1	2147483645	2147483645	(null)	65536	ONLINE
6	GE_DATA	8192	65536	(null)	1	2147483645	2147483645	(null)	65536	ONLINE
7	GE_IND	8192	65536	(null)	1	2147483645	2147483645	(null)	65536	ONLINE
8	UN_DATA	8192	65536	(null)	1	2147483645	2147483645	(null)	65536	ONLINE
9	UN_IND	8192	65536	(null)	1	2147483645	2147483645	(null)	65536	ONLINE

Mediante el SABD se puede observar el esquema UN creado:

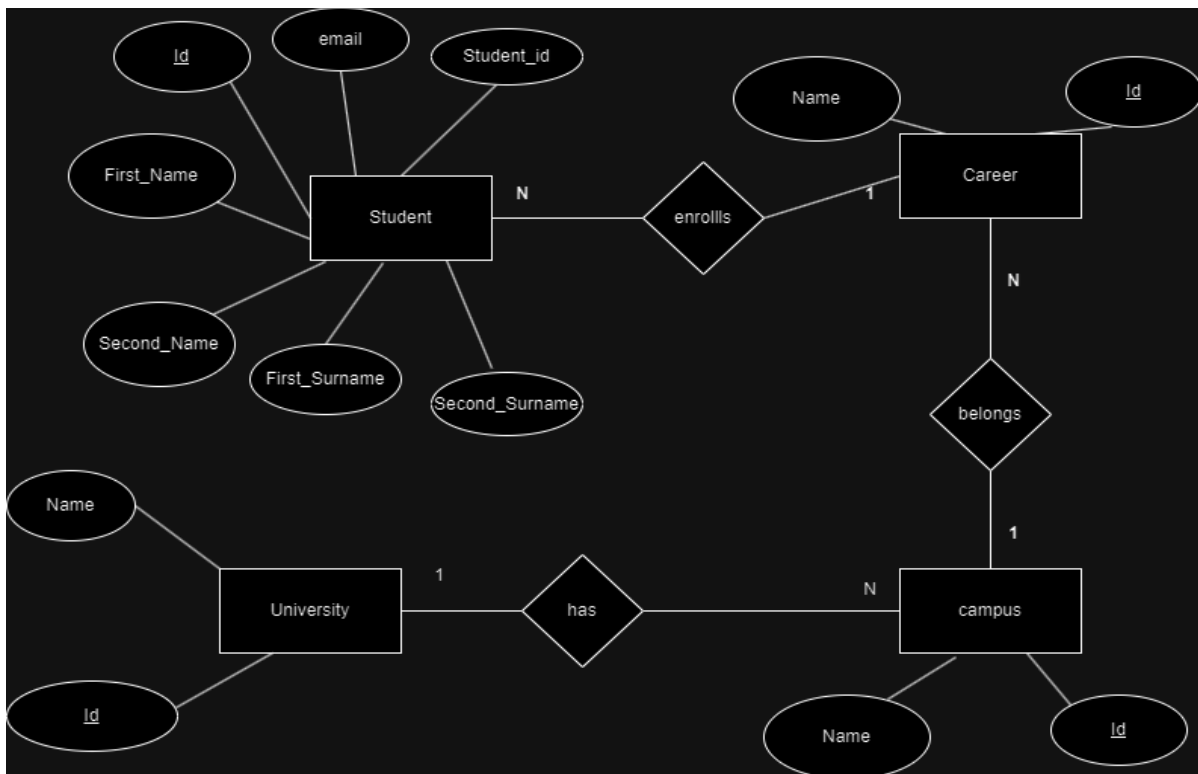


Ejercicio 2:

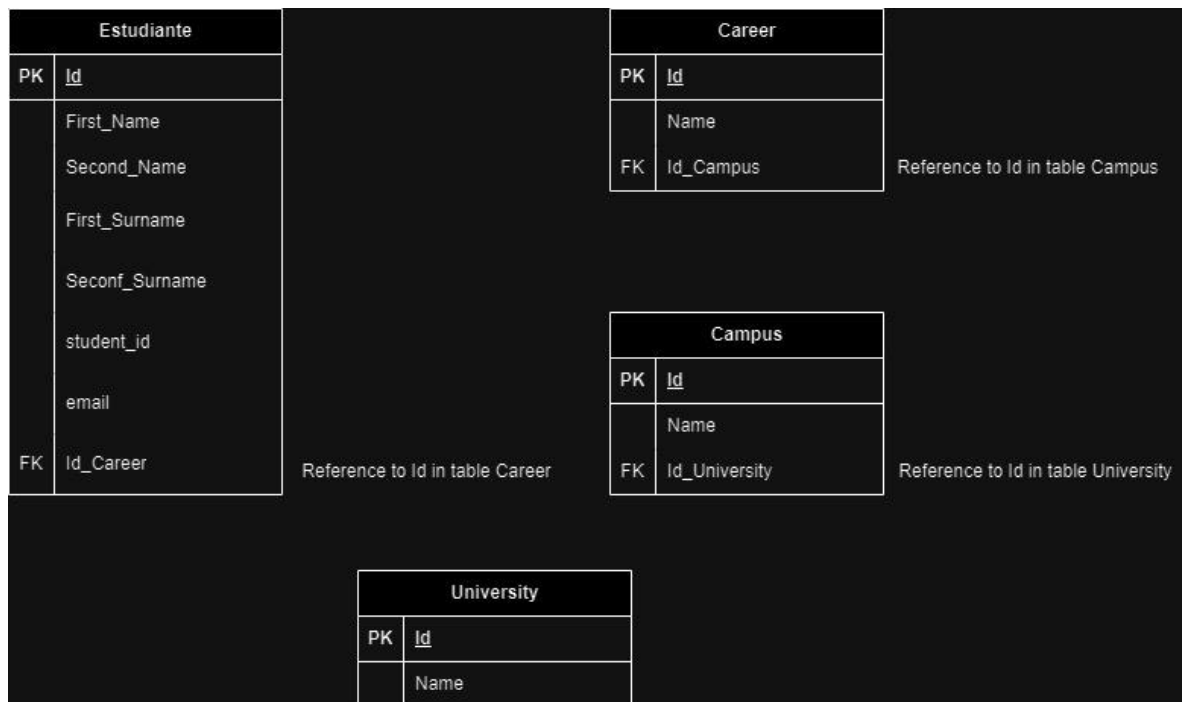
Desarrolle el modelo entidad-relación para crear un mini sistema de Universidad que incluya estudiantes, sede, carrera. Considere los siguientes requerimientos: 10pts

- a. Un estudiante sólo puede matricular una carrera
- b. Un estudiante sólo pertenece a una sede
- c. Una carrera sólo se imparte en una sede

Modelo lógico:



Modelo conceptual:



Ejercicio 3 y 4:

- Cree las llaves primarias y foráneas. Utilice secuencias para los campos llave.
- Cree todos los check constraints que considere necesario.

Creación de las tablas, University, Campus, Career y Student con sus respectivos PRIMARY KEY y Constraints.

```

1 DROP TABLE university;
2 CREATE TABLE UNIVERSITY(
3     id_university    NUMBER(6)    CONSTRAINT pk_university PRIMARY KEY,
4     university_name  VARCHAR2(25)  CONSTRAINT university_universityName_nn NOT NULL
5 );
  
```

```

1 DROP TABLE campus;
2 CREATE TABLE CAMPUS(
3     id_campus        NUMBER(6)    CONSTRAINT pk_campus PRIMARY KEY,
4     campus_name       VARCHAR2(25)  CONSTRAINT campus_campusName_nn NOT NULL,
5     id_university     NUMBER(6)
6 );
  
```

```

1 DROP TABLE career;
2 CREATE TABLE CAREER(
3     id_career      NUMBER(6)    CONSTRAINT pk_career PRIMARY KEY,
4     career_name    VARCHAR2(25) CONSTRAINT career_careerName_nn NOT NULL,
5     id_campus      NUMBER(6)
6 );

```

```

1 DROP TABLE student_seq;
2 CREATE TABLE STUDENT(
3     id_student     NUMBER(6)    CONSTRAINT pk_student PRIMARY KEY,
4     first_name     VARCHAR2(25) CONSTRAINT student_firstName_nn NOT NULL,
5     first_surname  VARCHAR2(25) CONSTRAINT student_firstSurname_nn NOT NULL,
6     second_name    VARCHAR2(25),
7     second_surname VARCHAR2(25) CONSTRAINT student_secondSurname_nn NOT NULL,
8     student_id     NUMBER(10)   CONSTRAINT student_sId_nn NOT NULL,
9     email          VARCHAR2(25) CONSTRAINT student_email_nn NOT NULL,
10    id_career       NUMBER(6)
11 );
12

```

Se le agrega el constraint de la respectiva FOREIGN KEY a cada tabla.

```

1 ALTER TABLE career
2 ADD
3 CONSTRAINT FK_CAMPUS_CAREER FOREIGN KEY (id_campus) REFERENCES campus(id_campus);
4
5 ALTER TABLE campus
6 ADD
7 CONSTRAINT FK_UNIVERSITY_CAMPUS FOREIGN KEY (id_university) REFERENCES university(id_university)
8
9 ALTER TABLE student
10 ADD
11 CONSTRAINT fk_career_student FOREIGN KEY (id_career) REFERENCES career(id_career);
12

```

Tablas:

- University:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_UNIVERSITY	NUMBER(6,0)	No	(null)	1	Primary key for University table
2	UNIVERSITY_NAME	VARCHAR2(25 BYTE)	No	(null)	2	University name

- Campus:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID_CAMPUS	NUMBER(6,0)	No	(null)	1	Primary key for Campus table
2	CAMPUS_NAME	VARCHAR2(25 BYTE)	No	(null)	2	Campus name
3	ID_UNIVERSITY	NUMBER(6,0)	Yes	(null)	3	Foreign Key, refers to Id in table University

- Career:

	❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1	ID_CAREER	NUMBER(6,0)	No	(null)		1 Primary key for Career table
2	CAREER_NAME	VARCHAR2(25 BYTE)	No	(null)		2 Careers name
3	ID_CAMPUS	NUMBER(6,0)	Yes	(null)		3 Foreign Key, references id in table campus

- Student:

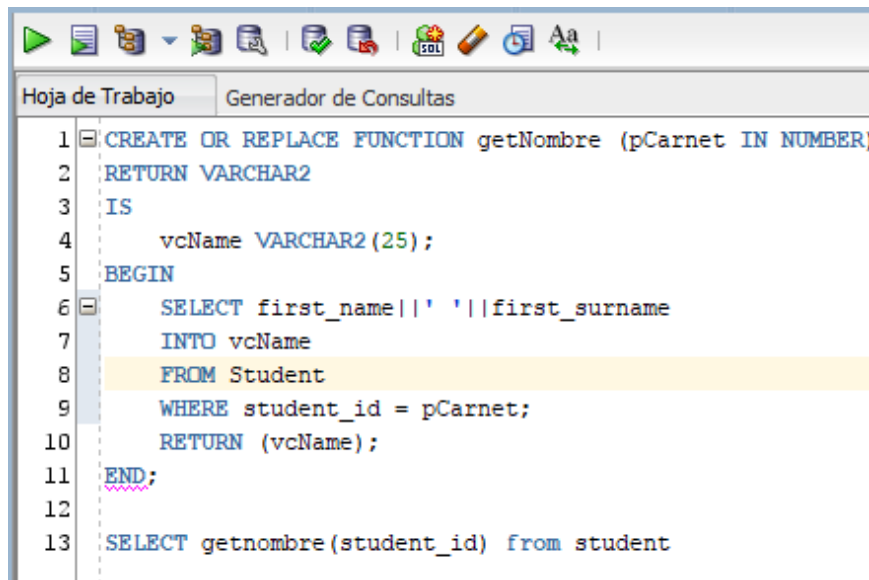
	❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1	ID_STUDENT	NUMBER(6,0)	No	(null)		1 Primary key for student table
2	FIRST_NAME	VARCHAR2(25 BYTE)	No	(null)		2 Student first name
3	FIRST_SURNAME	VARCHAR2(25 BYTE)	No	(null)		3 Student first surname
4	SECOND_NAME	VARCHAR2(25 BYTE)	Yes	(null)		4 Student second name
5	SECOND_SURNAME	VARCHAR2(25 BYTE)	No	(null)		5 Student second surname
6	STUDENT_ID	NUMBER(10,0)	No	(null)		6 Student's university id
7	EMAIL	VARCHAR2(25 BYTE)	No	(null)		7 Student's email
8	ID_CAREER	NUMBER(6,0)	Yes	(null)		8 Foreign key, references id in table career

Ejercicio 5:

Desarrolle las siguientes funciones, procedimientos o paquetes según corresponda:

- getNombre a partir del carnet.

Se crea el script de la función, este retorna el carnet junto al primer nombre y el primer apellido según los datos relacionados a cada carnet enviado por parámetro:



```

1 CREATE OR REPLACE FUNCTION getNombre (pCarnet IN NUMBER)
2 RETURN VARCHAR2
3 IS
4     vcName VARCHAR2(25);
5 BEGIN
6     SELECT first_name || ' ' || first_surname
7     INTO vcName
8     FROM Student
9     WHERE student_id = pCarnet;
10    RETURN (vcName);
11 END;
12
13 SELECT getnombre(student_id) from student
  
```


Retorna los nombres ordenados según los números de carnet registrados con el script “SELECT student_id, getnombre(student_id) from student”:

13

SELECT student_id,getnombre(student_id) from student

Salida de Script x

Resultado de la Consulta x

 Todas las Filas Recuperadas: 16 en 0,166 segundos

	STUDENT_ID	GETNOMBRE(STUDENT_ID)
1	2020147165	Juan Perez
2	2022352732	John Smith
3	2022619817	Michael Wilson
4	2022563471	Emily Miller
5	2021644362	David Taylor
6	2022155247	Christopher Martin
7	2022305894	Britney Walker
8	2023045856	Andrew White
9	2020191962	Olivia Jackson
10	2020917312	Emma Adams
11	2022565639	Brian Wright
12	2020709386	Isabella Hall
13	2020488170	James Moore
14	2021001505	Sophia Hernandez
15	2021910954	bree Gonzalez
16	2021247925	Jose Gomez

b. getCorreo a partir del id_estudiante

Se crea el script de la función, este retorna el id de estudiante junto al correo según los datos relacionados a cada id de estudiante enviado por parámetro:

```

1 CREATE OR REPLACE FUNCTION getCorreo (pID IN NUMBER)
2 RETURN VARCHAR2
3 IS
4     vcEmail VARCHAR2(25);
5 BEGIN
6     SELECT email
7     INTO vcEmail
8     FROM Student
9     WHERE id_student = pID;
10    RETURN (vcEmail);
11 END;
```

Retorna los correos ordenados según los números de id's registrados con el script "SELECT id_student, getCorreo(id_student) from student":

13

SELECT id_student, getCorreo(id_student) from student

Salida de Script x

Resultado de la Consulta x

SQL

Todas las Filas Recuperadas: 16 en 0,008 segundos

ID_STUDENT	GETCORREO(ID_STUDENT)
1	2 jperez@gmail.com
2	3 jsmith@gmail.com
3	4 mwilson@gmail.com
4	5 emiller@gmail.com
5	6 dtaylor@gmail.com
6	7 cmartin@gmail.com
7	8 bwalker@gmail.com
8	9 awhite@gmail.com
9	10 ojackson@gmail.com
10	11 leadams@gmail.com
11	12 bwright@gmail.com
12	13 ihall@gmail.com
13	14 jmoore@gmail.com
14	15 shernandez@gmail.com
15	16 bgonzales@gmail.com
16	17 jgomez@gmail.com

c. getCarrera a partir del id_estudiante

Se crea el script de la función, este retorna el id de estudiante junto al nombre de la carrera a la que pertenece según los datos relacionados a cada id de estudiante enviado por parámetro, esto se logra haciendo un join de las tablas Student y Career:

<div> <div>Hoja de Trabajo</div> <div>Generador de Consultas</div> </div>	
1	<code>CREATE OR REPLACE FUNCTION getCarrera (pID IN NUMBER)</code>
2	<code>RETURN VARCHAR2</code>
3	<code>IS</code>
4	<code> vcCarrera VARCHAR2 (25);</code>
5	<code>BEGIN</code>
6	<code> SELECT career_name</code>
7	<code> INTO vcCarrera</code>
8	<code> FROM Student a</code>
9	<code> JOIN career b on a.id_career = b.id_career</code>
10	<code> WHERE id_student = pID;</code>
11	<code> RETURN (vcCarrera);</code>
12	<code>END;</code>

Retorna los correos ordenados según los números de id's registrados con el script "SELECT id_student, getCarrera(id_student) from student":

14 | `SELECT id_student, getCarrera(id_student) from student`

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 16 en 0,017 segundos

ID_STUDENT	GETCARRERA(ID_STUDENT)
1	2 Ing.Computacion
2	3 Ing.Computacion
3	4 Ing.Forestal
4	5 Ing.Forestal
5	6 Ing.Prod.Industrial
6	7 Ing.Prod.Industrial
7	8 Admin.Empresas
8	9 Admin.Empresas
9	10 Criminología
10	11 Criminología
11	12 Enseñanza. Matematicas
12	13 Enseñanza. Matematicas
13	14 Enseñanza.Est.Sociales
14	15 Enseñanza.Est.Sociales
15	16 Animación digital
16	17 Animación digital

d. getSede a partir del id_estudiante

Se crea el script de la función, este retorna el id de estudiante junto al nombre de la carrera a la que pertenece según los datos relacionados a cada id de estudiante enviado por parámetro, esto se logra haciendo un join de las tablas Student y Career y luego otro con las tablas Career y Campus:

```

1 CREATE OR REPLACE FUNCTION getSede (pID IN NUMBER)
2 RETURN VARCHAR2
3 IS
4     vcSede VARCHAR2(25);
5 BEGIN
6     SELECT campus_name
7     INTO vcSede
8     FROM Student a
9     JOIN career b on a.id_career = b.id_career
10    JOIN Campus c on b.id_campus = c.id_campus
11    WHERE id_student = pID;
12    RETURN (vcSede);
13 END;
```


Retorna los correos ordenados según los números de id's registrados con el script "SELECT id_student, getSede(id_student) from student":

```
15 | SELECT id_student, getSede(id_student) from student
```

Salida de Script x		Resultado de la Consulta x	
		Todas las Filas Recuperadas: 16 en 0,017 segundos	
ID_STUDENT	GETSEDE(ID_STUDENT)		
1	2 San José		
2	3 San José		
3	4 San José		
4	5 San José		
5	6 Cartago		
6	7 Cartago		
7	8 Cartago		
8	9 Cartago		
9	10 San Pedro		
10	11 San Pedro		
11	12 San Pedro		
12	13 San Pedro		
13	14 Alajuela		
14	15 Alajuela		
15	16 Alajuela		
16	17 Alajuela		