



ABP BASE DE DATOS

INTEGRANTES

Jesús David Montiel Vergara

Perez Arrieta Boris Yesid

PROFESOR

Jhon Carlos Arrieta

CARTAGENA, BOLÍVAR

22/04/2021

INTRODUCCIÓN	5
JUSTIFICACIÓN	5
OBJETIVOS GENERALES	5
OBJETIVOS ESPECÍFICOS	5
ENUNCIADO	6
DIAGRAMAS DER Y MR	7
Creación de base de datos en MySQL	8
Diagrama del resultado	11
Creación de base de datos en PostgreSQL	12
Diagrama del resultado	14
Creación de base de datos en SQL Server	15
Diagrama del resultado	17
Creación de base de datos en ORACLE	18
Copia de seguridad en MySQL	20
Copia de seguridad en SQL Server	22
Copia de seguridad en PostgreSQL	24
Creación de vistas	27

¿ Por qué usar Vistas ?	27
MySQL:	27
Oracle:	28
PostgreSQL:	29
SQL Server:	29
Creación de vistas en MySQL	30
Creación de vistas en SQL Server	35
Creación de vistas en SQL Server	38
Creación de transacciones en MySQL con Java	41
Creamos los query:	41
Creamos la transacción:	42
Código completo:	45
Usuarios en MySQL	47
Crear:	47
Privilegios:	47
Eliminar:	48
Eliminar privilegios:	49
Usuarios en SQL Server	50
Crear:	50
Privilegios:	50
Eliminar:	51
Usuarios en PostgreSQL	52

Crear:	52
Usuarios en ORACLE	54
Crear:	54
Privilegios:	54
Eliminar:	54
Eliminar privilegios:	55
Exportar un archivo Excel a una base de datos MySQL	56
Sentencias para consultas MySQL	62
Exportar un archivo Excel a una base de datos SQL Server	68
Sentencias para consultas para SQL Server	82
Exportar un archivo Excel a una base de datos PostgreSQL	84
Sentencias para consultas para PostgreSQL	91
Exportar un archivo Excel a una base de datos Oracle	93
Sentencias para consultas en ORACLE	100
Repositorio GitHub	101

INTRODUCCIÓN

En el siguiente documento nos encontraremos con 4 motores de base de datos donde se le aplicaran consultas en los diferentes motores de base de datos, también nos encontraremos con la creación de vistas, transacciones, diagrama relaciones, diagrama entidad relación y los resultados obtenidos de la investigación

JUSTIFICACIÓN

Después de hacer varias investigaciones en diferentes motores de base de datos para poder brindarle a los lectores que tengan este documento a la mano facilitarle la búsqueda en los motores de base de datos como MySQL, Oracle, PostgreSQL y SQLServer, desde la creación de una base de datos hasta la implementación de transacciones en las diferentes base de datos.

OBJETIVOS GENERALES

- Definir los conceptos de los 4 motores de base de datos
- Afianzar los conocimientos de los usuarios finales

OBJETIVOS ESPECÍFICOS

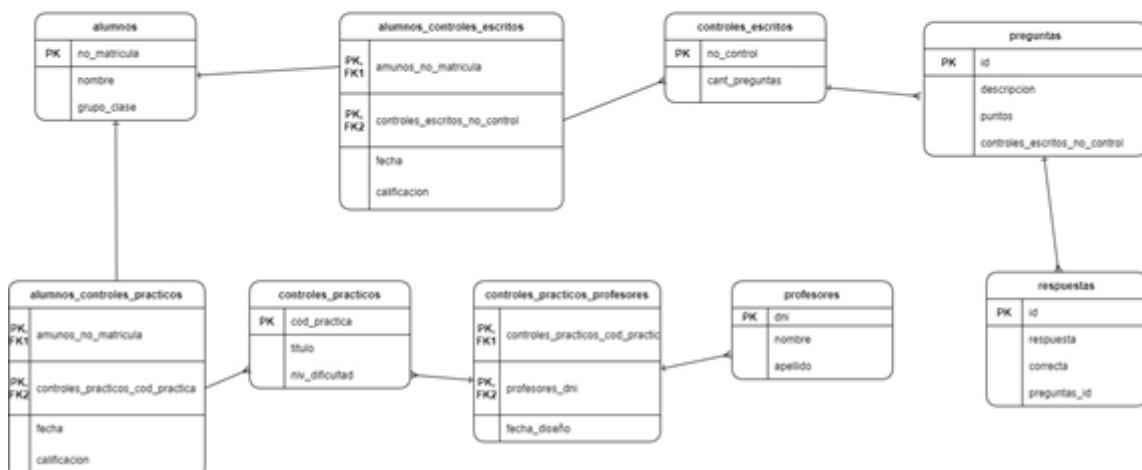
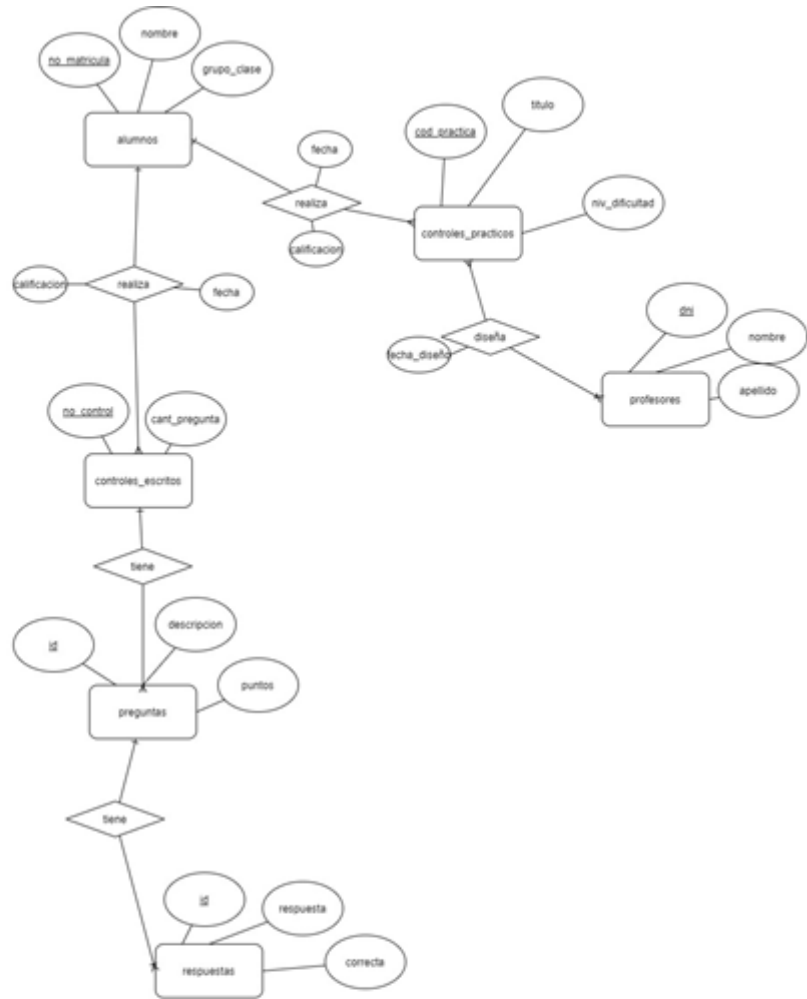
- Crear base de datos
- Consultas empresariales
- Implementar transacciones
- Interpretar los DER

ENUNCIADO

Los profesores de la asignatura de Computación II de la Universidad de Chile deciden crear una base de datos que contenga la información de los resultados de los controles realizadas a los alumnos. Para realizar el diseño se sabe que:

- Los alumnos están definidos por su n° de matrícula, nombre y el grupo al que asisten a clase. Dichos alumnos realizan dos tipos de controles a lo largo del curso académico:
- Controles escritos: cada alumno realiza varios a lo largo del curso, y se definen por el n° de control, el n° de preguntas de que consta y la fecha de realización (la misma para todos los alumnos que realizan el mismo control). Evidentemente, es importante almacenar la nota de cada alumno por control.
- Prácticas: se realiza un n° indeterminado de ellas durante el curso académico, algunas serán en grupo y otras individuales. Se definen por un código de práctica, título y el grado de dificultad. En este caso los alumnos pueden examinarse de cualquier práctica cuando lo deseen, debiéndose almacenar la fecha y nota obtenida.
- En cuanto a los profesores, únicamente interesa conocer (además de sus datos personales: DNI y nombre), quien es el que ha diseñado cada práctica, sabiendo que en el diseño de una práctica puede colaborar más de uno, y que un profesor puede diseñar más de una práctica. Interesa, además, la fecha en que ha sido diseñada cada práctica por el profesor correspondiente

DIAGRAMAS DER Y MR



Creación de base de datos en MySQL

Creamos la base de datos con el siguiente comando:

```
CREATE DATABASE examenes;
```

Resultado:

```
mysql> CREATE DATABASE examenes;  
Query OK, 1 row affected (0.00 sec)  
  
mysql>
```

Nos establecemos en la base de datos que acabamos de crear:

```
USE examenes;
```

Resultado:

```
mysql> USE examenes;  
Database changed  
mysql> _
```


Ahora creamos las tablas con el comando `CREATE TABLE` y el nombre de la tabla, seguido de todas las tablas que esta contendrá, su tipo de dato, si acepta nulo o no y si será alguna llave, y al finalizar se utiliza el comando `ENGINE = INNODB` para poder relacionar las tablas más adelante.

```
CREATE TABLE alumnos (  
  no_matricula INT NOT NULL PRIMARY KEY,  
  nombre VARCHAR(30) NOT NULL,  
  grupo_clase VARCHAR(30) NOT NULL) ENGINE = INNODB;
```

Resultado:

```
mysql> CREATE TABLE profesores(  
  -> dni INT(15) NOT NULL PRIMARY KEY,  
  -> nombre VARCHAR(20) NOT NULL,  
  -> apellido VARCHAR(20) NOT NULL  
  -> )ENGINE = INNODB;  
Query OK, 0 rows affected (0.20 sec)
```

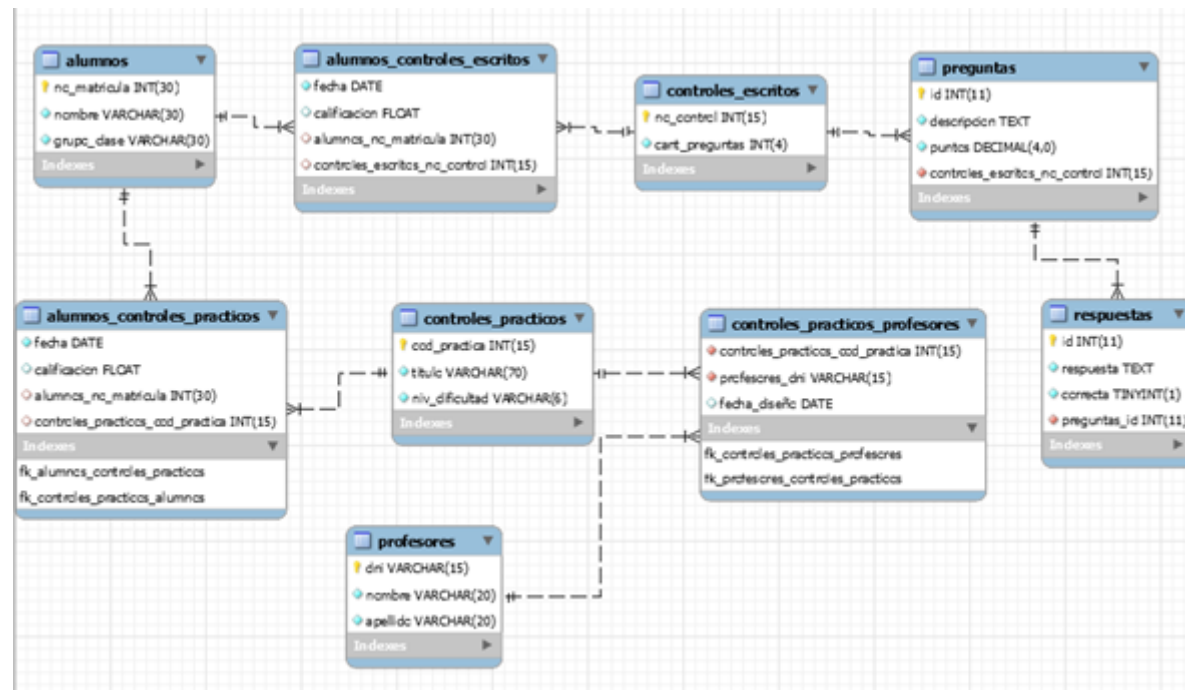
Ahora agregamos las llaves foraneas con el comando *ALTER TABLE* y el nombre de la tabla en la que se aplicará seguido del comando *ADD CONSTRAINT* y le colocamos el nombre de la llave foranea seguido de *FOREIGN KEY* y se especifica el nombre y por último el comando *REFERENCES* para referenciar la tabla con su nombre y dentro de paréntesis la llave primaria que será referenciada.

```
ALTER TABLE name_table
ADD CONSTRAINT fk_name
FOREIGN KEY(name_foreign_key)
REFERENCES reference_table(primary_key);
```

Resultado:

```
mysql> ALTER TABLE respuestas
-> ADD CONSTRAINT fk_preguntas_respuestas
-> FOREIGN KEY(preguntas_id)
-> REFERENCES preguntas(id);
Query OK, 0 rows affected (0.91 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Diagrama del resultado



Creación de base de datos en PostgreSQL

Creamos la base de datos con el siguiente comando:

```
CREATE DATABASE examenes;
```

Resultado:

```
postgres=# CREATE DATABASE examenes;  
CREATE DATABASE  
postgres=#  
postgres=# _
```

Ahora creamos las tablas con el comando CREATE TABLE y el nombre de la tabla, seguido de todas las tablas que esta contendrá, su tipo de dato, si acepta nulo o no y si será alguna llave.

```
CREATE TABLE name_table(  
  name_row TYPE NULL/NOT NULL,  
  name_row TYPE NULL/NOT NULL,  
  name_row TYPE NULL/NOT NULL,  
  name_row TYPE NULL/NOT NULL);
```

Resultado:

```
CREATE TABLE
exámenes=# CREATE TABLE controles_practicos_profesores(
exámenes=# controles_practicos_cod_practica INT,
exámenes=# profesores_dni VARCHAR(15),
exámenes=# fecha_diseno DATE);
```

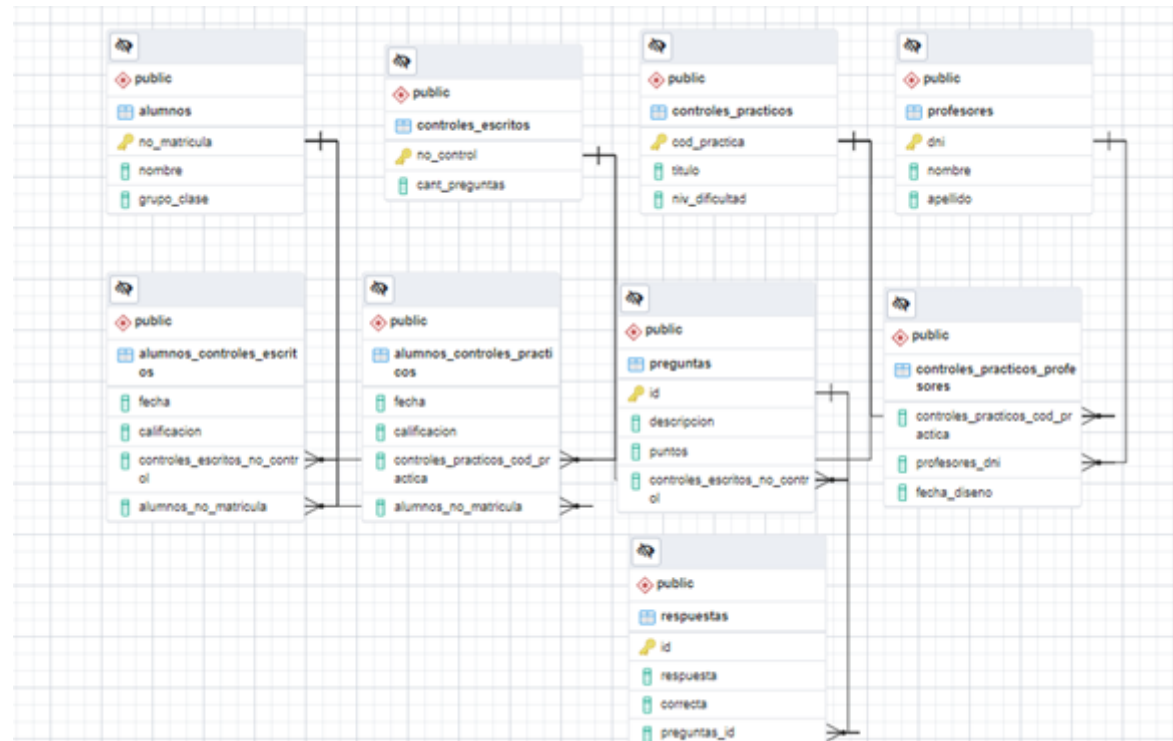
Ahora agregamos las llaves foraneas con el comando *ALTER TABLE* y el nombre de la tabla en la que se aplicará seguido del comando *ADD CONSTRAINT* y le colocamos el nombre de la llave foranea seguido de *FOREIGN KEY* y se especifica el nombre y por último el comando *REFERENCES* para referenciar la tabla con su nombre y dentro de paréntesis la llave primaria que será referenciada.

```
ALTER TABLE name_table
ADD CONSTRAINT fk_name
FOREIGN KEY(name_foreign_key)
REFERENCES reference_table(primary_key);
```

Resultado:

```
exámenes=# ALTER TABLE controles_practicos_profesores(
ALTER TABLE
exámenes=# ALTER TABLE controles_practicos_profesores
exámenes=# ADD CONSTRAINT fk_profesores_controles_practicos
exámenes=# FOREIGN KEY (profesores_dni)
exámenes=# REFERENCES profesores(dni)
exámenes=# ;
ALTER TABLE
exámenes=#
```

Diagrama del resultado



Creación de base de datos en SQL Server

Creamos la base de datos con el siguiente comando:

```
CREATE DATABASE examenes;
```

Resultado:

```
2> CREATE DATABASE examenes
3> go
1> USE examenes
```

Ahora creamos las tablas con el comando CREATE TABLE y el nombre de la tabla, seguido de todas las tablas que esta contendrá, su tipo de dato, si acepta nulo o no y si será alguna llave.

```
CREATE TABLE name_table(
name_row TYPE NULL/NOT NULL,
name_row TYPE NULL/NOT NULL,
name_row TYPE NULL/NOT NULL,
name_row TYPE NULL/NOT NULL);
```

Resultado:

```
1> CREATE TABLE preguntas(
2> id INT NOT NULL PRIMARY KEY,
3> descripcion VARCHAR(30) NOT NULL,
4> puntos FLOAT NOT NULL,
5> controles_escritos_no_control INT
6> )
7>
```

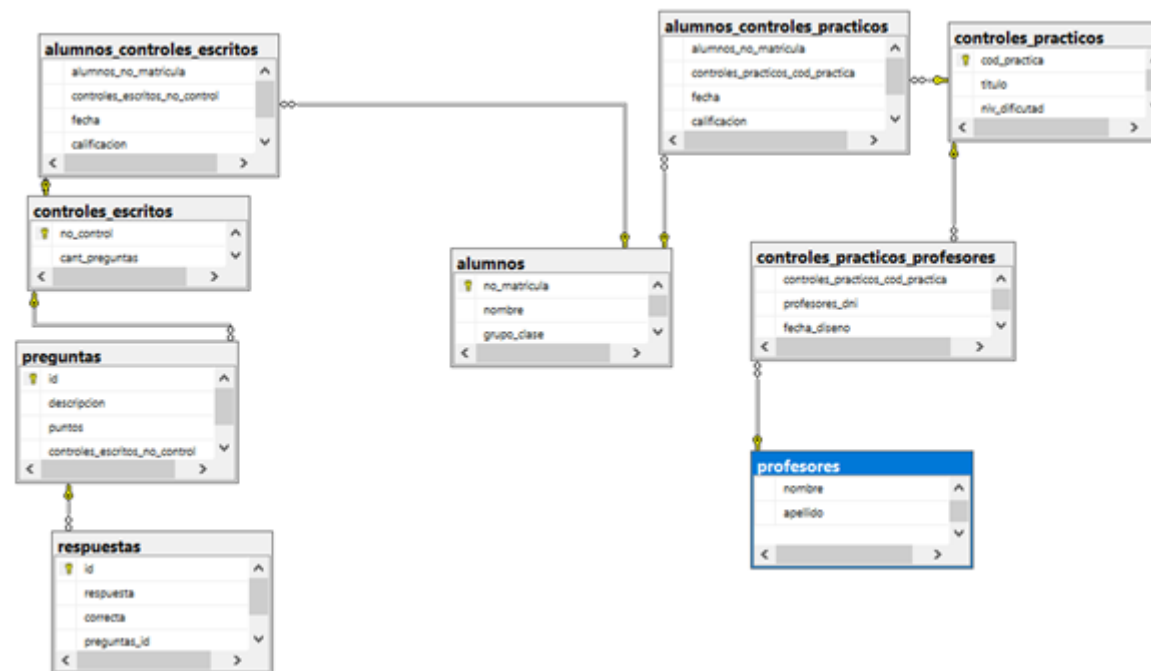
Ahora agregamos las llaves foraneas con el comando *ALTER TABLE* y el nombre de la tabla en la que se aplicará seguido del comando *ADD CONSTRAINT* y le colocamos el nombre de la llave foranea seguido de *FOREIGN KEY* y se especifica el nombre y por último el comando *REFERENCES* para referenciar la tabla con su nombre y dentro de paréntesis la llave primaria que será referenciada.

```
ALTER TABLE name_table
ADD CONSTRAINT fk_name
FOREIGN KEY(name_foreign_key)
REFERENCES reference_table(primary_key);
```

Resultado:

```
1> ALTER TABLE alumnos_controles_escritos
2> ADD CONSTRAINT fk_controles_escritos_alumnos
3> FOREIGN KEY(controles_escritos_no_control)
4> REFERENCES controles_escritos(no_control)
5> go
```


Diagrama del resultado



Creación de base de datos en ORACLE

Ahora creamos las tablas con el comando CREATE TABLE y el nombre de la tabla, seguido de todas las tablas que esta contendrá, su tipo de dato, si acepta nulo o no y si será alguna llave, y al finalizar se utiliza el comando *ENGINE = INNODB* para poder relacionar las tablas más adelante.

```
CREATE TABLE alumnos (  
  no_matricula INT NOT NULL PRIMARY KEY,  
  nombre VARCHAR(30) NOT NULL,  
  grupo_clase VARCHAR(30) NOT NULL) ENGINE = INNODB;
```

Resultado:

```
SQL> CREATE TABLE alumnos (  
  2 no_matricula INT NOT NULL PRIMARY KEY,  
  3 nombre VARCHAR(30) NOT NULL,  
  4 grupo_clase VARCHAR(30) NOT NULL  
  5 );
```

Tabla creada.

Ahora agregamos las llaves foraneas con el comando *ALTER TABLE* y el nombre de la tabla en la que se aplicará seguido del comando *ADD CONSTRAINT* y le colocamos el nombre de la llave foranea seguido de *FOREIGN KEY* y se especifica el nombre y por último el comando *REFERENCES* para referenciar la tabla con su nombre y dentro de paréntesis la llave primaria que será referenciada.

```
ALTER TABLE name_table
ADD CONSTRAINT fk_name
FOREIGN KEY(name_foreign_key)
REFERENCES reference_table(primary_key);
```

Resultado:

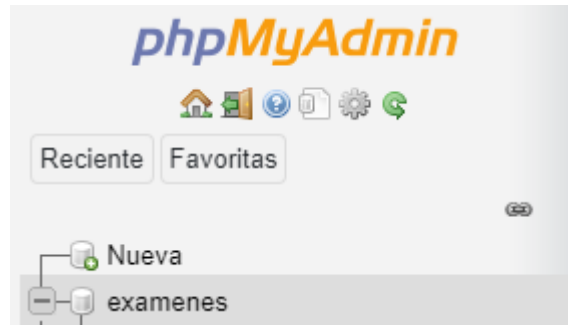
```
SQL> ALTER TABLE alumnos_controles_escritos
2  ADD CONSTRAINT fk_controles_escritos_alumnos
3  FOREIGN KEY(controles_escritos_no_control)
4  REFERENCES controles_escritos(no_control);

Tabla modificada.
```

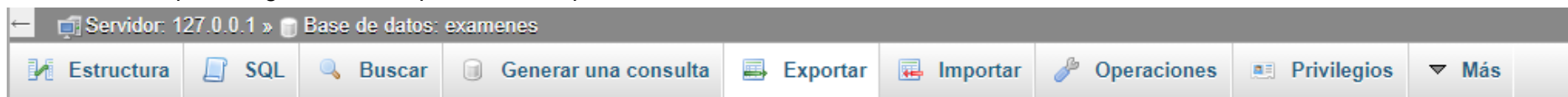
Copia de seguridad en MySQL

Para realizar una copia de seguridad en MySQL a través de phpMyAdmin.

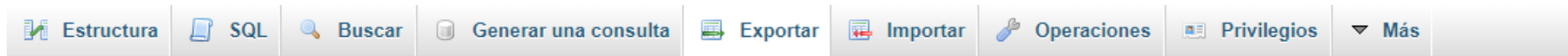
Entramos a nuestra base de datos que en este caso es exámenes.



En el banner superior ingresamos al apartado de Exportar.



En el método de exportación seleccionamos Rápido y posteriormente le damos clic en el botón continuar.



Exportando tablas de la base de datos "exámenes"

Exportar plantillas:

Nueva plantilla:

Plantillas existentes:

Plantilla:

Método de exportación:

- ☒ Rápido - mostrar sólo el mínimo de opciones de configuración
- ☐ Personalizado - mostrar todas las opciones de configuración posibles

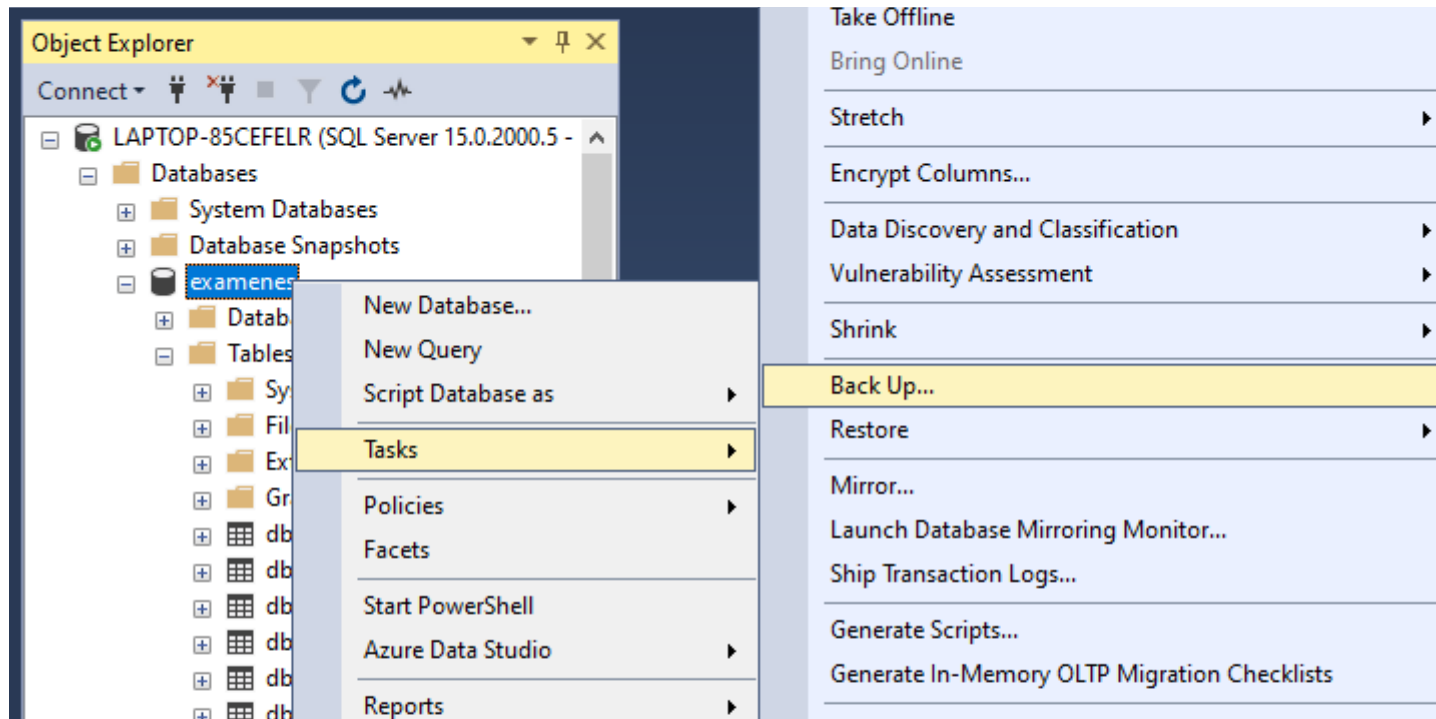
Formato:



Copia de seguridad en SQL Server

Para realizar una copia de seguridad en SQL Server a través de Microsoft SQL Server Management Studio.

Entramos a nuestra base de datos que en este caso es exámenes y le damos clic derecho -> Tasks -> Back Up...



Se nos abre esta ventana y solamente le daremos clic en OK.

The screenshot shows the 'Backup' wizard in SQL Server Enterprise Manager. The window is divided into a left sidebar and a main configuration area.

Left Sidebar:

- Connection:** Server: LAPTOP-85CEFELR, Connection: LAPTOP-85CEFELR\boris. A link 'View connection properties' is present.
- Progress:** A circular progress indicator and the text 'Ready'.

Main Configuration Area:

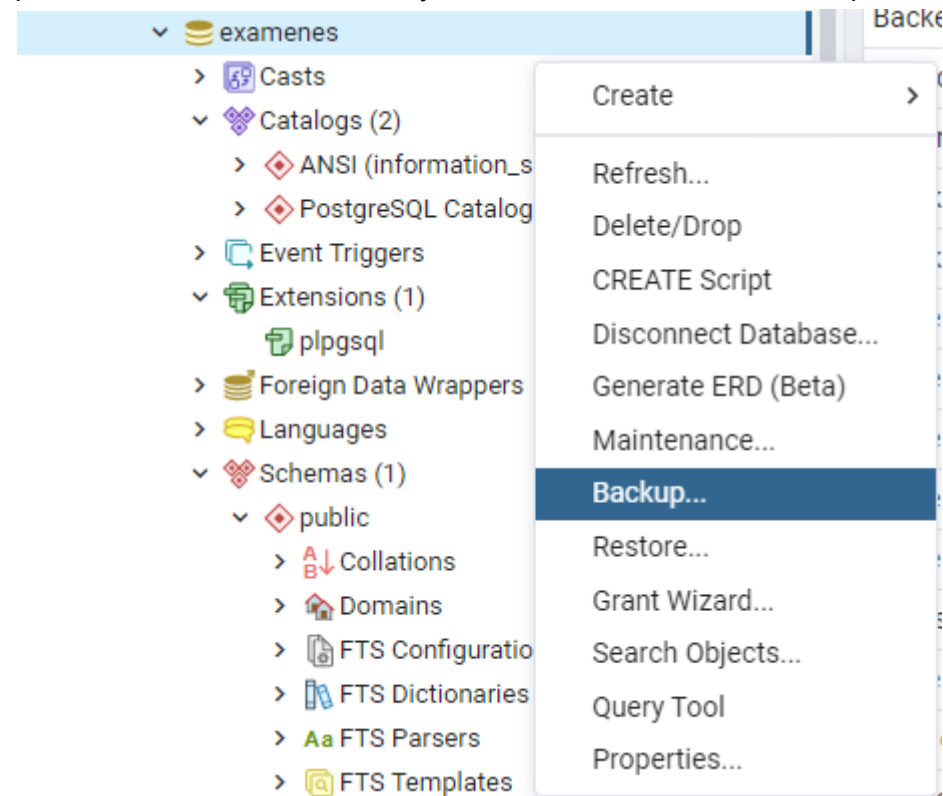
- Backup type:** A dropdown menu set to 'Full'.
- Copy-only backup:** An unchecked checkbox.
- Backup component:** Two radio buttons: 'Database' (selected) and 'Files and filegroups' (unselected). A text box and a browse button '...' are next to the 'Files and filegroups' option.
- Destination:** A section header.
- Back up to:** A dropdown menu set to 'Disk'.
- Backup list:** A list box containing the path 'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\examenes.bak'. To the right of the list are three buttons: 'Add...', 'Remove', and 'Contents'.

Bottom: 'OK' and 'Cancel' buttons.

Copia de seguridad en PostgreSQL

Para realizar una copia de seguridad en PostgreSQL a través de pgAdmin 4.

Entramos a nuestra base de datos que en este caso es exámenes y le damos clic derecho -> Backup...



En la ventana que se nos abre en:

- Filename: Seleccionamos la ruta y nombre del archivo que se va a crear.
- Encoding: UTF8

Los demás campos se pueden dejar por default. Al finalizar dar clic en el botón Backup.

Backup (Database: examenes)

General | Dump options

Filename: C:\Users\boris\Downloads\examenes.sql

Format: Custom

Compression ratio:

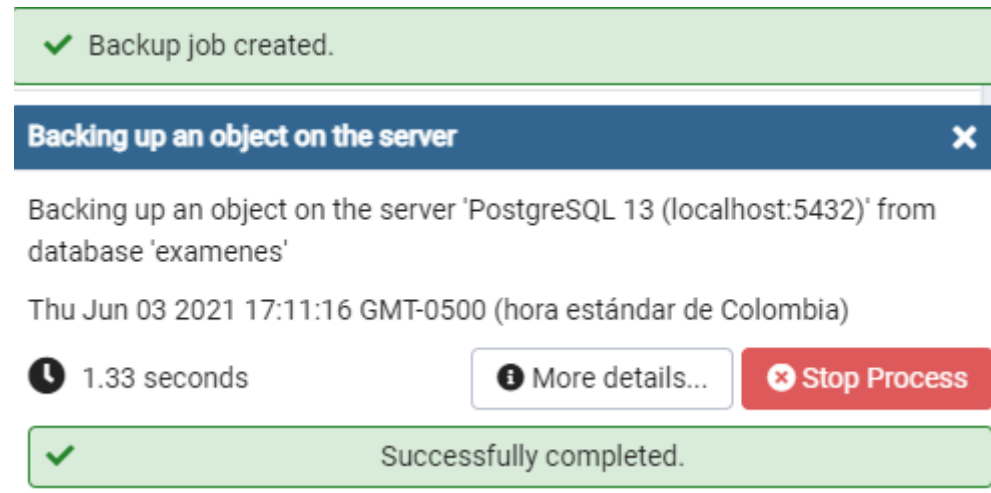
Encoding: UTF8

Number of jobs:

Role name: postgres

Cancel **Backup**

Si todo salió bien en la ventana de alerta que se abre en la parte inferior derecha de la pantalla debe salir el mensaje de Successfully completed.

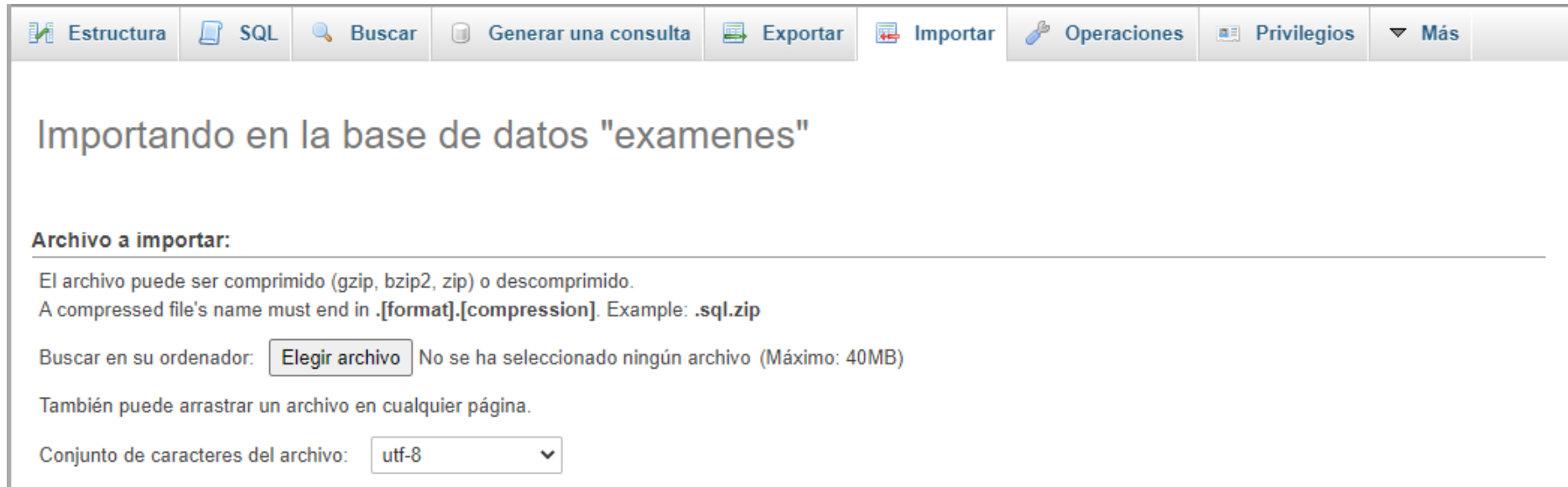


Restaurar una copia de seguridad en MySQL

Para restaurar una copia de seguridad en MySQL utilizando phpMyAdmin entramos a nuestra base de datos y en el banner superior damos clic a importar



En el apartado de Archivo a importar le indicaremos el archivo que vamos a restaurar, clic en Elegir archivo y buscarlo.



Una vez seleccionado lo demás lo dejamos por defecto y le damos clic en continuar.

Estructura	SQL	Buscar	Generar una consulta	Exportar	Importar	Operaciones	Privilegios	Más
------------	-----	--------	----------------------	----------	----------	-------------	-------------	-----

Importación parcial:

☒

Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to import large files, however it can break transactions.)*

Omitir esta cantidad de consultas (en SQL) desde la primera:

Otras opciones:

☒ Habilite la revisión de las claves foráneas

Formato:

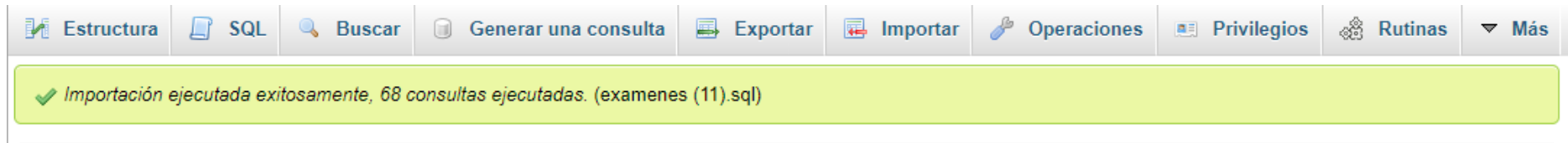
Opciones específicas al formato:

Modalidad SQL compatible:

☒ No utilizar AUTO_INCREMENT con el valor 0

[Continuar](#)

Una vez terminada la ejecución nos debe mostrar este mensaje de confirmación que el proceso se hizo correctamente.

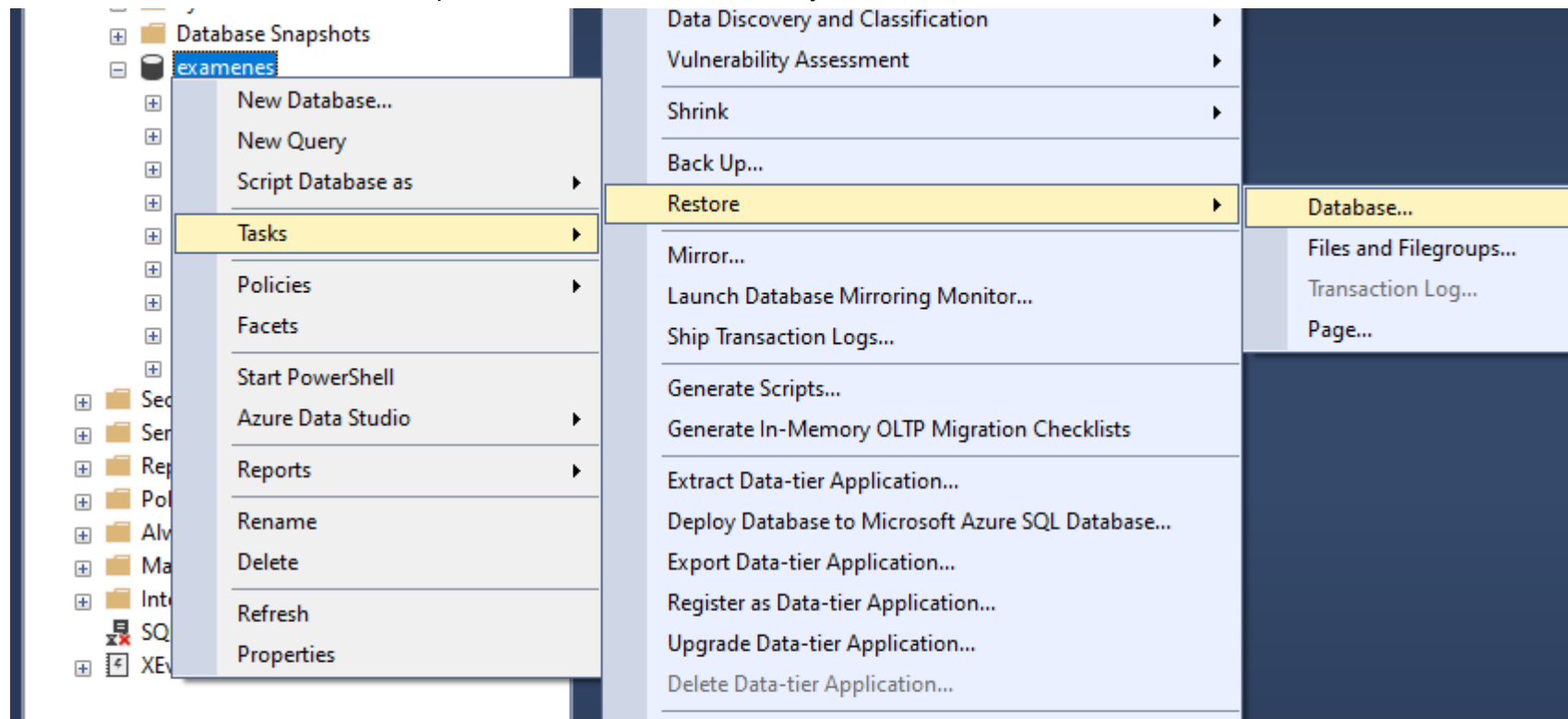


The screenshot shows a horizontal menu bar with the following items: Estructura, SQL, Buscar, Generar una consulta, Exportar, Importar, Operaciones, Privilegios, Rutinas, and Más. Below the menu bar is a green message box with a checkmark icon and the text: "Importación ejecutada exitosamente, 68 consultas ejecutadas. (exámenes (11).sql)".

Restaurar una copia de seguridad en SQL Server

Para restaurar una copia de seguridad en SQL Server a través de Microsoft SQL Server Management Studio.

Entramos a nuestra base de datos que en este caso es exámenes y le damos clic derecho -> Tasks -> Restore -> Database...



En la ventana que se nos abre, podemos ver las copias de seguridad que habíamos hecho y para restaurar nada más debemos seleccionar la que queremos y dar clic en Ok

Script | ? Help

Source

☒ Database: exámenes

☐ Device:

Database:

Destination

Database: exámenes

Restore to: The last backup taken (jueves, junio 03, 2021 4:40:52 PM) Timeline...

Restore plan

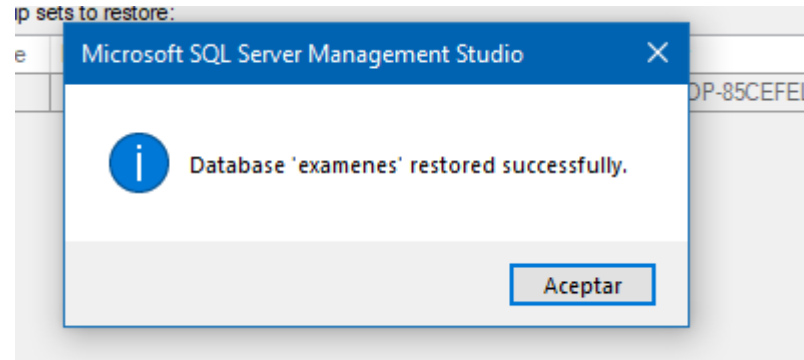
Backup sets to restore:

Restore	Name	Component	Type	Server	Database	Position	First LSN
<input checked="" type="checkbox"/>	exámenes-Full Database Backup	Database	Full	LAPTOP-85CEFELR	exámenes	1	4000000

Verify Backup Media

OK Cancel Help

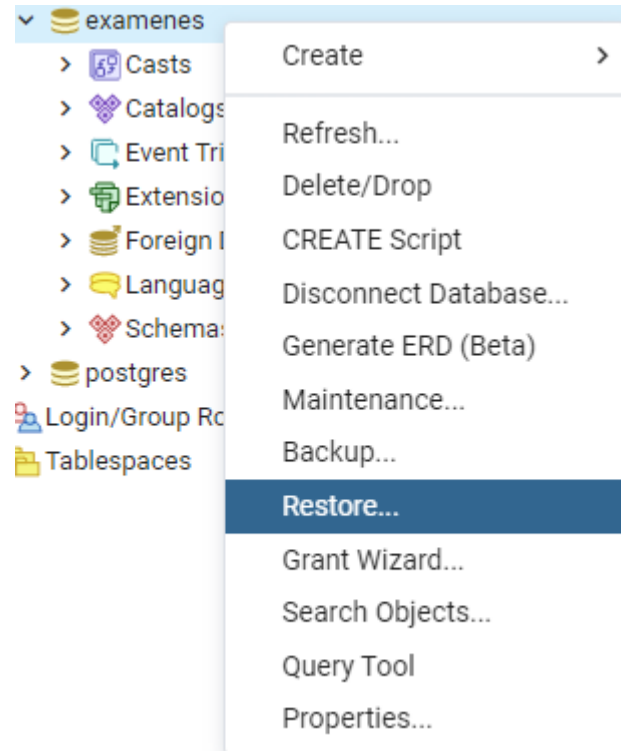
Si todo el proceso se realizó correctamente, nos mostrará este mensaje de confirmación.



Restaurar una copia de seguridad de PostgreSQL

Para restaurar una copia de seguridad en PostgreSQL a través de pgAdmin 4.

Entramos a nuestra base de datos que en este caso es exámenes y le damos clic derecho -> Restore...



En la ventana que se nos abre buscamos el archivo a restaurar en el apartado de Filename y una vez seleccionado le damos en Restore.

The image shows a screenshot of the 'Restore (Database: examenes)' dialog box in PostgreSQL. The dialog has a title bar with the text 'Restore (Database: examenes)' and a close button. Below the title bar, there are two tabs: 'General' and 'Restore options'. The 'General' tab is currently selected. It contains four fields: 'Format' with a dropdown menu showing 'Custom or tar', 'Filename' with a text box containing 'C:\Users\boris\Downloads\examenes.sql' and a browse button (three dots), 'Number of jobs' with an empty text box, and 'Role name' with a dropdown menu showing 'postgres' and a user icon. At the bottom of the dialog, there are three buttons: an information button (i), a help button (?), and a 'Cancel' button. To the right of these is a 'Restore' button with an upward arrow icon. The 'Restore' button is highlighted in blue.

Creación de vistas

Una vista es una tabla virtual basada en una tabla u otra vista, no contiene datos en sí misma, pero es como una ventana a través de la cual se pueden ver o cambiar los datos de las tablas. Podemos representar con ellas subconjuntos lógicos o combinación de datos, las tablas sobre las cuales se basa una vista se llaman tablas base.

¿ Por qué usar Vistas ?

- Para restringir el acceso a la B.D.
- Para realizar consultas complejas de manera fácil.
- Para obtener una independencia de los datos.
- Para presentar diferentes vistas de los mismos datos.

MySQL:

Creación: Se crea una vista con todos las filas y columnas de la *tabla_basada*.

```
CREATE VIEW VistaAccesoAS SELECT * FROM acceso;
```

Actualizar: A diferencia de la anterior, en esta utilizamos *CREATE OR REPLACE VIEW*

```
CREATE OR REPLACE VIEW nombre_vista_crear AS SELECT * FROM tabla_basada;
```

Eliminar: Para eliminar una vista simplemente se utiliza el comando *DROP VIEW* y el nombre de la vista:

```
DROP VIEW nombre_vista_crear;
```

Oracle:

Creación: Crear una vista sobre la tabla que uno la quiere hacer;

```
CREATE VIEW nombre_vista_crear  
AS  
  
SELECT nombre,  
        apellidos,  
        matricula  
FROM tabla_basada;
```

ALTER VIEW: Si queremos modificar la definición de la vista se utiliza la sentencia ALTER VIEW, de forma muy parecida a como se hacían con las tablas.

```
ALTER VIEW nombre_vista_crear  
AS  
(  
SELECT nombre,  
        apellidos,  
        matricula,  
        fx_alquiler,  
        fx_devolucion
```

```
FROM tabla_basada;
```

Eliminar: También podemos eliminar las vistas a través de la sentencia `DROP VIEW`. para eliminar la vista que hemos creado.

```
DROP VIEW nombre_vista_creada;
```

PostgreSQL:

Crear: Para crear una vista en este motor es totalmente igual a MySQL. Respetando el estándar básico de SQL

```
CREATE VIEW nombre_vista_crear AS SELECT * FROM tabla_basada;
```

Eliminar: También podemos eliminar las vistas a través de la sentencia `DROP VIEW`. para eliminar la vista que hemos creado.

```
DROP VIEW nombre_vista_creada;
```

SQL Server:

Creación :Podemos crear la vista usando la sentencia `CREATE VIEW`. Una vista se puede crear desde una sola tabla o varias tablas.

```
CREATE VIEW view_name AS  
SELECT column1, column2.....  
FROM table_name  
WHERE condition;
```

Creación de vistas en MySQL

```
CREATE VIEW vistaAcceso AS SELECT * FROM acceso; --Creación de la vista  
SELECT * FROM vistaAcceso; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

CT * FROM vistaAcesso x

Page Size: 50 | Total Rows: 47 Page: 1 of 1

codigo	nombre
100070012	ADRIANA CAROLINA HERNANDEZ MONTERROZA
100070013	ADRIANA MARCELA REY SANCHEZ
100070014	ALEJANDRO ABONDANO ACEVEDO
100070015	ALEJANDRO ABONDANO ACEVEDO
100070016	ANDREA CATALINA ACERO CARO
100070017	ANDREA LILIANA CRUZ GARCIA
100070018	ANDRES FELIPE VILLA MONROY
100070019	ANGELA PATRICIA MAHECHA PI?EROS
100070020	ANGELICA LISSETH BLANCO CONCHA
100070021	ANGELICA MARIA ROCHA GARCIA
100070022	ANGIE TATIANA FERN?NDEZ MART?NEZ
100070023	TATIANA ANGIE MART?NEZ FERN?NDEZ
100070024	CAMILO VILLAMIZAR ARISTIZABAL

```
CREATE VIEW vistaAlumnos AS SELECT * FROM alumnos; --Creación de la vista
SELECT * FROM vistaAlumnos; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

ECT * FROM vistaAlumno... x

Page Size: 50 | Total Rows: 24 | Page: 1 of 1

no_matricula	nombre	grupo_clase
100070012	ADRIANA CAROLINA HERNANDEZ MON	ING_SIST 128CSIS-1
100070013	ADRIANA MARCELA REY SANCHEZ	ING_IND 128CSIS-2
100070014	ALEJANDRO ABONDANO ACEVEDO	ING_CVL 128CSIS-3
100070015	ALEJANDRO ABONDANO ACEVEDO	ING_SIST 128CSIS-1
100070016	ANDREA CATALINA ACERO CARO	ING_SIST 128CSIS-1
100070017	ANDREA LILIANA CRUZ GARCIA	ING_SIST 128CSIS-1
100070018	ANDRES FELIPE VILLA MONROY	ING_SIST 128CSIS-1
100070019	ANGELA PATRICIA MAHECHA PI?ERO	ING_SIST 128CSIS-1
100070020	ANGELICA LISSETH BLANCO CONCHA	ING_SIST 128CSIS-1
100070021	ANGELICA MARIA ROCHA GARCIA	ING_SIST 128CSIS-1

```
CREATE VIEW vistaAlumnosControlesEscritos AS SELECT * FROM alumnos_controles_escritos; --Creación de la vista
SELECT * FROM vistaAlumnosControlesEscritos; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

SELECT * FROM vistaAlumno... x

Page Size: 50 | Total Rows: 12 | Page: 1 of 1 | Matching Rows:

#	fecha	calificacion	alumnos_no_matricula	controles_escritos_no_control
1	2021-05-15	4.0	100070012	7000
2	2021-05-15	4.9	100070013	15037
3	2021-05-17	2.0	100070014	15039
4	2021-05-18	4.0	100070015	15040
5	2021-05-15	4.3	100070016	15041
6	2021-05-15	4.1	100070017	15042
7	2021-05-15	4.0	100070018	15043
8	2021-05-22	5.0	100070019	15044
9	2021-05-18	4.0	100070020	15045
10	2021-05-18	4.6	100070021	15046
11	2021-05-13	3.0	100070012	15039
12	2021-05-12	1.5	100070012	15041

```
CREATE VIEW vistaAlumnosControlesPracticos AS SELECT * FROM alumnos_controles_practicos; --Creación de la vista
SELECT * FROM vistaAlumnosControlesPracticos; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

SELECT * FROM vistaAlumno...

Page Size: 50 | Total Rows: 6 | Page: 1 of 1 | Matching Rows:

fecha	calificacion	alumnos_no_matricula	controles_practicos_cod_practica
2021-05-31	2.0	100070028	9741
2021-06-01	2.0	100070029	14790
2021-06-02	3.0	100070030	3456789
2021-06-03	3.0	100070031	12345678
2021-06-04	3.0	100070032	87654321
2021-05-08	4.2	100070012	9741

```
CREATE VIEW vistaProfesores AS SELECT * FROM profesores; --Creación de la vista
SELECT * FROM vistaProfesores; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

SELECT * FROM vistaProfes... x

Page Size: 50 | Total Rows: 24 | Page: 1 of 1

dni	nombre	apellido
UNI-1020	M?NICA ALEXANDRA	CAMACHO AMAYA
UNI-1021	JOS? GUILLERMO	MARIN ZUBIETA
UNI-1022	HUGO ANDR?S	CAMARGO VARGAS
UNI-1023	INGRID ROCIO	GUERRERO PENAGOS
UNI-1024	IV?N DAVID	CORAL BURBANO
UNI-1025	IVONNE JOULIETTE	BARRERA LOPEZ
UNI-1026	JENNY FERNANDA	S?NCHEZ ARENAS
UNI-1027	JENNY VIVIANA	MONCALEANO PRECIADO
UNI-1028	JORGE ESTEBAN	REY BOTERO
UNI-1029	JORGE MARIO	OROZCO DUSS?N
UNI-1030	M?NICA NATALIA	CAMARGO MENDOZA
UNI-1031	NATALIA ANDREA	GUTI?RREZ VELASCO
UNI-1032	NATALIA MELISSA	BARRERO FORERO
UNI-1033	NATALIA VIVY	CASAS P?EZ
UNI-1034	OLGA VIVIANA	OVALLE SOLANO
UNI-1035	OSCAR DAVID	COLMENARES BARBUDO
UNI-1036	OSCAR JULIAN	ULLOA ORJUELA
UNI-1037	PABLO	URIBE ANTIA
UNI-1038	SANDRA XIMENA	ALVAREZ CASTILLO
UNI-1039	SANDRA XIMENA	SANDRA XIMENA
UNI-1040	SEBASTIAN	BORDA MELGUIZO

Creación de vistas en SQL Server

```
CREATE VIEW vistaAlumnos AS SELECT * FROM alumnos; --Creación de la vista
```

Resultado:

Messages

Commands completed successfully.

Completion time: 2021-05-21T13:45:42.5843842-05:00

```
SELECT * FROM vistaAlumnos; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

	no_matricula	nombre	grupo_clase
1	100070012	ADRIANA CAROLINA HERNANDEZ MONTERROZA	ING_SIST 128CSIS-1
2	100070013	ADRIANA MARCELA REY SANCHEZ	ING_IND 128CSIS-2
3	100070014	ALEJANDRO ABONDANO ACEVEDO	ING_CVL 128CSIS-3
4	100070015	ALEJANDRO ABONDANO ACEVEDO	ING_SIST 128CSIS-1
5	100070016	ANDREA CATALINA ACERO CARO	ING_SIST 128CSIS-1
6	100070017	ANDREA LILIANA CRUZ GARCIA	ING_SIST 128CSIS-1
7	100070018	ANDRES FELIPE VILLA MONROY	ING_SIST 128CSIS-1
8	100070019	ANGELA PATRICIA MAHECHA PIÑEROS	ING_SIST 128CSIS-1
9	100070020	ANGELICA LISSETH BLANCO CONCHA	ING_SIST 128CSIS-1
10	100070021	ANGELICA MARIA ROCHA GARCIA	ING_SIST 128CSIS-1
11	100070022	ANGIE TATIANA FERNÁNDEZ MARTÍNEZ	ING_SIST 128CSIS-1

```
CREATE VIEW vistaAcceso AS SELECT * FROM acceso; --Creación de la vista
```

Resultado:

Messages

Commands completed successfully.

Completion time: 2021-05-21T13:45:42.5843842-05:00

```
SELECT * FROM vistaAcceso; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

	codigo	nombre	pass	rol
1	100070012	ADRIANA CAROLINA HERNANDEZ MONTERROZA	12345	alumno
2	100070013	ADRIANA MARCELA REY SANCHEZ	12346	alumno
3	100070014	ALEJANDRO ABONDANO ACEVEDO	12347	alumno
4	100070015	ALEJANDRO ABONDANO ACEVEDO	12348	alumno
5	100070016	ANDREA CATALINA ACERO CARO	12349	alumno
6	100070017	ANDREA LILIANA CRUZ GARCIA	12350	alumno
7	100070018	ANDRES FELIPE VILLA MONROY	12351	alumno
8	100070019	ANGELA PATRICIA MAHECHA PIÑEROS	12352	alumno
9	100070020	ANGELICA LISSETH BLANCO CONCHA	12353	alumno
10	100070021	ANGELICA MARIA ROCHA GARCIA	12354	alumno
11	100070022	ANGIE TATIANA FERNÁNDEZ MARTÍNEZ	12355	alumno

Creación de vistas en PostgreSQL

```
CREATE VIEW vistaProfesoresAS SELECT * FROM profesores; --Creación de la vista
```

Resultado:

Data Output Explain Messages Notifications

CREATE VIEW

Query returned successfully in 473 msec.

```
SELECT * FROM vistaAcceso; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

Data Output Explain Messages Notifications

	dni character varying (15)	nombre character varying (30)	apellido character varying (30)	
1	UNI-1020	MànICA ALEXANDRA	CAMACHO AMAYA	
2	UNI-1021	JOSE GUILLERMO	MARIN ZUBIETA	
3	UNI-1022	HUGO ANDRÉS	CAMARGO VARGAS	
4	UNI-1023	INGRID ROCIO	GUERRERO PENAGOS	
5	UNI-1024	IVpN DAVID	CORAL BURBANO	

Universidad

Transacciones

Una transacción es una unidad de trabajo compuesta por diversas tareas, cuyo resultado final debe ser que se ejecuten todas o ninguna de ellas. Una transacción tiene 4 propiedades fundamentales que son:

Atomicidad: Significa que una transacción es una unidad indivisible, es decir que se ejecuta o no se ejecuta.

Creación de transacciones en MySQL con Java

Creamos los query:

```
String insertarAlumnos = ("INSERT INTO alumnos(no_matricula, nombre, grupo_clase) VALUES ('" + noMatricula + "', '" + nombre + "', '" + grupoClase + "')");
String insertarAcceso = ("INSERT INTO acceso(codigo, nombre, password, rol) "
    + "VALUES ('" + noMatricula + "', '" + nombre + "', '" + noMatricula + "', 'alumno')");
String insertarCalificacionControlEscrito = ("INSERT INTO alumnos_controles_escritos(fecha, calificacion, alumnos_no_matricula, controles_escritos_no_control) "
    + "VALUES ('" + fecha + "', '" + calificacionEscrito + "', '" + noMatricula + "', '" + noControlEscrito + "')");
String insertarCalificacionControlPractico = ("INSERT INTO alumnos_controles_practicos(fecha, calificacion, alumnos_no_matricula, controles_practicos_cod_practica) "
    + "VALUES ('" + fecha + "', '" + calificacionPractico + "', '" + noMatricula + "', '" + noControlPractico + "')");
```

Creamos la transacción:

En java, insertamos dentro de un try catch toda nuestra transacción:

```
try {
    miConexion = DriverManager.getConnection(url, user, pass); //Creamos la conexión con la BD.

    miConexion.setAutoCommit(false);
    Statement miStatement = miConexion.createStatement();

    miStatement.executeUpdate(insertarAlumnos); //Pasamos el query de insertar alumnos creado anteriormente.
    miStatement.executeUpdate(insertarAcceso); //Pasamos el query de insertar acceso creado anteriormente.
    miStatement.executeUpdate(insertarCalificacionControlEscrito); //Pasamos el query de insertar acceso calificaciones escritas.
    miStatement.executeUpdate(insertarCalificacionControlPractico); //Pasamos el query de insertar calificaciones practicas.
    miConexion.commit(); // Utilizamos este comando para guardar todos los cambios.
    resultado = "Todo Ok"; //Si no se encuentra ningún problema nos arroja como resultado "Todo ok".

} catch (SQLException e) {
    resultado = "Ocurrió un error, no se hizo"; //En caso de que haya algún problema nos mostrará este mensaje y NO se realiza la consulta.
    System.out.println("ERROR: " + e);

    if (miConexion != null) {
        try {
            miConexion.rollback(); //Usamos rollback() para revertir todos los cambios que se hubieran hecho en la transacción.
        } catch (SQLException ex) {
            Logger.getLogger(conexion.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

```
}  
}  
}
```

Por defecto, MySQL se ejecuta en modo autocommit. Esto significa que tan pronto como se ejecuta una sentencia se actualiza (modifica) la tabla. Para evitar lo anterior utilizamos el siguiente comando:

```
miConexion.setAutoCommit(false);
```

```

try {
    miConexion = DriverManager.getConnection(url, user, pass);

    miConexion.setAutoCommit(false);
    Statement miStatement = miConexion.createStatement();

    miStatement.executeUpdate(insertarAlumnos);
    miStatement.executeUpdate(insertarAcceso);
    miStatement.executeUpdate(insertarCalificacionControlEscrito);
    miStatement.executeUpdate(insertarCalificacionControlPractico);
    miConexion.commit();
    resultado = "Todo Ok";
} catch (SQLException e) {
    resultado = "Ocurrió un error, no se hizo";
    System.out.println("ERROR: " + e);

    if (miConexion != null) {
        try {
            miConexion.rollback();
        } catch (SQLException ex) {
            Logger.getLogger(conexion.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
}

```

Código completo:

```
public String registrarAlumno(int noMatricula, String nombre, String grupoClase, String fecha, float
calificacionEscrito, float calificacionPractico, int noControlEscrito, int noControlPractico) {
    String resultado;
    Connection miConexion = null;
    String insertarAlumnos = ("INSERT INTO alumnos(no_matricula, nombre, grupo_clase) VALUES ('" + noMatricula +
    "',' + nombre + "',' + grupoClase + "')");
    String insertarAcceso = ("INSERT INTO acceso(codigo, nombre, password, rol) VALUES ('" + noMatricula + "','
    "' + nombre + "',' + noMatricula + "',' alumno')");
    String insertarCalificacionControlEscrito = ("INSERT INTO alumnos_controles_escritos(fecha, calificacion,
    alumnos_no_matricula, controles_escritos_no_control) VALUES ('" + fecha + "',' + calificacionEscrito + "',' +
    noMatricula + "',' + noControlEscrito + "')");
    String insertarCalificacionControlPractico = ("INSERT INTO alumnos_controles_practicos(fecha, calificacion,
    alumnos_no_matricula, controles_practicos_cod_practica) VALUES ('" + fecha + "',' + calificacionPractico + "',' +
    noMatricula + "',' + noControlPractico + "')");

    try {
        miConexion = DriverManager.getConnection(url, user, pass);

        miConexion.setAutoCommit(false);
        Statement miStatement = miConexion.createStatement();

        miStatement.executeUpdate(insertarAlumnos);
        miStatement.executeUpdate(insertarAcceso);
        miStatement.executeUpdate(insertarCalificacionControlEscrito);
    }
}
```

```
        miStatement.executeUpdate(insertarCalificacionControlPractico);
        miConexion.commit();
        resultado = "Todo Ok";
    } catch (SQLException e) {
        resultado = "Ocurrió un error, no se hizo";
        System.out.println("ERROR:  " + e);

        if (miConexion != null) {
            try {
                miConexion.rollback();
            } catch (SQLException ex) {
                Logger.getLogger(conexion.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

    return resultado;
}
}
```

Usuarios en MySQL

Crear:

La sentencia para crear un usuario en MySQL es la siguiente:

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'password';
```

Con el comando “CREATE USER” le indicamos que queremos crear un usuario y seguido con comillas simples (‘’) se indica el nombre del usuario a crear, en este caso el nombre fue: ‘JM-BP’ seguido de un arroba (@) y se indica el nombre del servidor igualmente dentro de comillas simples como en nuestro caso que fue ‘localhost’ y seguido el comando “IDENTIFIED BY” para indicarle la contraseña que queremos establecer la cual la debemos escribir dentro de comillas simples como en nuestro ejemplo que fue: ‘123456’.

```
CREATE USER 'JM-BP'@'localhost' IDENTIFIED BY '123456';
```

Resultado:

```
mysql> create user 'JM-BP' identified by '123456';  
Query OK, 0 rows affected (0.07 sec)
```

Privilegios:

Ahora recordar que solamente se creó el usuario, falta establecer privilegios a este.

Para ello utilizamos el comando “GRANT” y en nuestro ejemplo lo seguimos con “ALL” lo cual indica que va a tener todos los privilegios (SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ETC) en la tabla que le indiquemos con el comando “PRIVILEGES ON” y seguido el nombre de la base de datos en que queremos establecer esos privilegios, en nuestro caso nada más elegimos la base de datos *exámenes*. En caso de querer que estos privilegios se apliquen en todas las bases de datos, solamente se coloca el símbolo asterisco (*) y seguido un punto (.) y a continuación en qué tablas, en nuestro ejemplo lo colocamos para todas las tablas con el símbolo asterisco (*) y “TO” para especificar a qué usuario es al que se lo vamos a aplicar, en este caso al mismo que creamos en el punto anterior.

```
GRANT ALL PRIVILEGES ON examenes.* TO 'JM-BP'@'localhost';
```

Resultado:

```
mysql> grant all privileges on examenes.* to 'JM-BP'@'localhost';  
Query OK, 0 rows affected (0.02 sec)
```

Por último queda aplicar el comando “FLUSH PRIVILEGES” para refrescar los privilegios.

```
FLUSH PRIVILEGES;
```

Resultado:

```
mysql> flush privileges;  
Query OK, 0 rows affected (0.08 sec)
```

Eliminar:

Para eliminar un usuario simplemente utilizamos el siguiente comando donde indicamos el nombre del usuario y nombre del servidor. En nuestro ejemplo utilizamos el usuario ‘JM-BP’ y el servidor ‘localhost’.


```
DROP USER 'JM-BP'@'localhost';
```

Resultado:

```
MariaDB [(none)]> DROP USER 'JM-BP'@'localhost';  
Query OK, 0 rows affected (0.067 sec)
```

Eliminar privilegios:

Para eliminar privilegios utilizamos el comando “REVOKE” seguido del “ALL” que cumple la misma función de la explicada al momento de establecer los privilegios. Luego con “PRIVILEGES ON” le indicamos si queremos quitar los privilegios en todas las bases de datos y tablas como en el ejemplo con los símbolos asterisco (*) sabiendo que el primer asteriscos bases de datos y el segundo asterisco tablas seguido del “FROM” e indicamos el nombre del usuario y servidor donde aplicaremos la revocación de privilegios.

```
REVOKE ALL PRIVILEGES ON *.* FROM 'JM-BP'@'localhost';
```

Resultado:

```
MariaDB [(none)]> REVOKE ALL PRIVILEGES ON *.* FROM 'JM-BP'@'localhost';  
Query OK, 0 rows affected (0.069 sec)
```

Usuarios en SQL Server

Crear:

Para crear un usuario en SQL Server se utiliza el comando CREATE LOGIN seguido del nombre del usuario que en este ejemplo es: 'jmbp' y WITH PASSWORD para establecer la contraseña que en este caso sería: '123456'.

```
CREATE LOGIN jmbp WITH PASSWORD = '123456';
```

Privilegios:

Ahora solo falta agregar los privilegios.

Debemos ingresar a la base de datos donde queremos otorgarle los permisos:

```
USE examenes;
```

Utilizamos el comando GRANT y los privilegios que le daremos, en este ejemplo serán todos entonces por ello le sigue: ALL PRIVILEGES TO y se escribe el nombre del usuario que creamos en el punto anterior que sería jmbp.

```
GRANT ALL PRIVILEGES TO jmbp;
```

Eliminar:

Para eliminar un usuario simplemente utilizamos el siguiente comando donde indicamos el nombre del usuario. En nuestro ejemplo utilizamos el usuario jmbp.

```
DROP USER jmbp;
```

Usuarios en PostgreSQL

Crear:

Para crear el usuario en postgresQL utilizamos la siguiente instrucción CREATE USER seguido del nombre del usuario a crear y colocarle la contraseña WITH PASSWORD y se le coloca la contraseña que desee en este caso se utiliza '123456'.

```
CREATE USER JMBP with password '123456';
```

Resultado:

```
postgres=# create user JMBP with password '123456';
CREATE ROLE
```

Para visualizar los usuarios creado colocamos \du y vemos que el usuario creado anteriormente no tiene ningún privilegio

Resultado:

Nombre de rol	Atributos
jmbp	
postgres	Superusuario, Crear rol, Crear BD, Replicación, Ignora RLS

Para agregarle un privilegio a un usuario colocamos ALTER user seguido del usuario que se creó anteriormente en este caso JMBP dándole el privilegio de superusuario colocando WITH SUPERUSER;

```
ALTER USER JMBP WITH SUPERUSER;
```

Resultado:

```
postgres=# ALTER USER JMBP with superuser;
ALTER ROLE
```

Para darle privilegios al usuario creado al principio JMBP a una base de datos es parecido al motor de base de datos MySQL usamos GRANT ALL ON DATABASE seguido del nombre de la base de datos en este caso exámenes en el usuario creado TO JMBP

```
GRANT ALL ON DATABASE exámenes TO JMBP;
```

Resultado:

```
postgres=# GRANT ALL ON DATABASE exámenes to JMBP;  
GRANT  
postgres=#
```

Usuarios en ORACLE

Crear:

Para crear un usuario en ORACLE se utiliza el comando CREATE LOGIN seguido del nombre del usuario que en este ejemplo es: 'jmbp' y IDENTIFIED BY para establecer la contraseña que en este caso sería: 'contras'.

```
CREATE USER jmbp IDENTIFIED BY contras;
```

Privilegios:

Para agregar privilegios a este usuario utilizamos el comando GRANT y los privilegios que le daremos, en este ejemplo serán todos entonces por ello le sigue: ALL PRIVILEGES TO y se escribe el nombre del usuario que creamos en el punto anterior que sería jmbp.

```
GRANT ALL PRIVILEGES TO jmbp;
```

Eliminar:

Para eliminar un usuario simplemente utilizamos el siguiente comando donde indicamos el nombre del usuario. En nuestro ejemplo utilizamos el usuario jmbp.

```
DROP USER jmbp;
```

Eliminar privilegios:

Para quitar los privilegios al usuario creado anteriormente utilizaremos el comando “REVOKE” seguido del “ALL” que cumple la misma función de la explicada al momento de establecer los privilegios. Luego con “PRIVILEGES FROM” donde le diremos a cual usuario le eliminaremos los privilegios.

```
REVOKE ALL PRIVILEGES FROM jmbp;
```

Exportar un archivo Excel a una base de datos MySQL

Para exportar un Archivo Excel a una base de datos MySQL, luego de hacer la respectiva limpieza y estructuración de los datos realizamos los siguientes pasos:

1. Se guarda el documento con el mismo nombre de la tabla en MySQL con el formato CSV (MS-DOS) como se evidencia en la *figura número 1*.
2. Vamos a MySQL y entramos a nuestra tabla donde insertamos los datos y en el banner superior se entra al apartado: "Importar" como se evidencia en la *figura número 2*.
3. Una vez dentro, se subirá el archivo .CSV que se guardó en el paso 1 haciendo clic en elegir archivo como se evidencia en la *figura 3*.
4. Seleccionamos el archivo .CSV creado en el paso 1.
5. Antes de dar a continuar, nos dirigimos a la parte inferior en las Opciones específicas al formato y en: "Columnas separadas por: "
6. Una vez de haber importados todos los datos del documento .CSV podemos visualizar que se han insertado de una manera satisfactoria

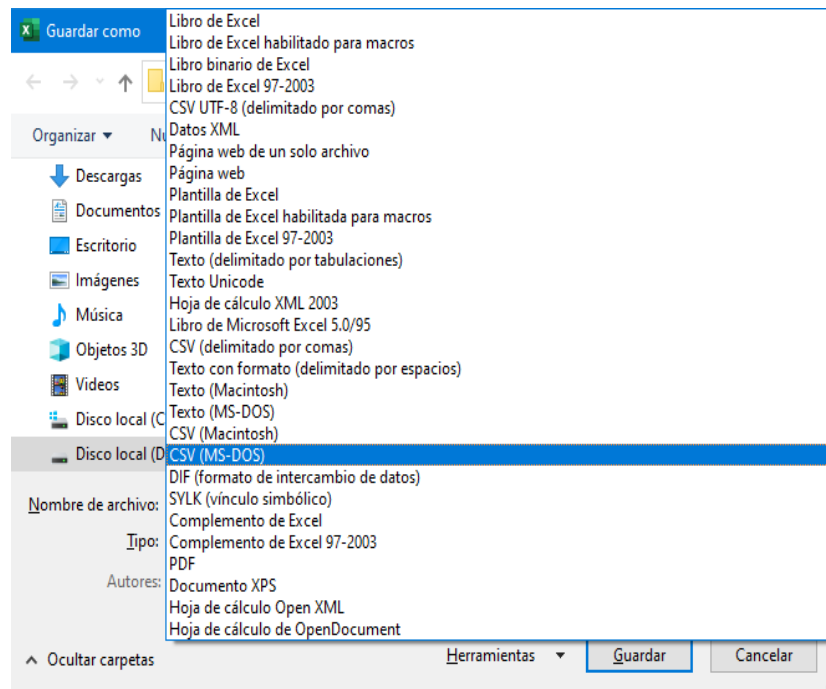


Figura 1.

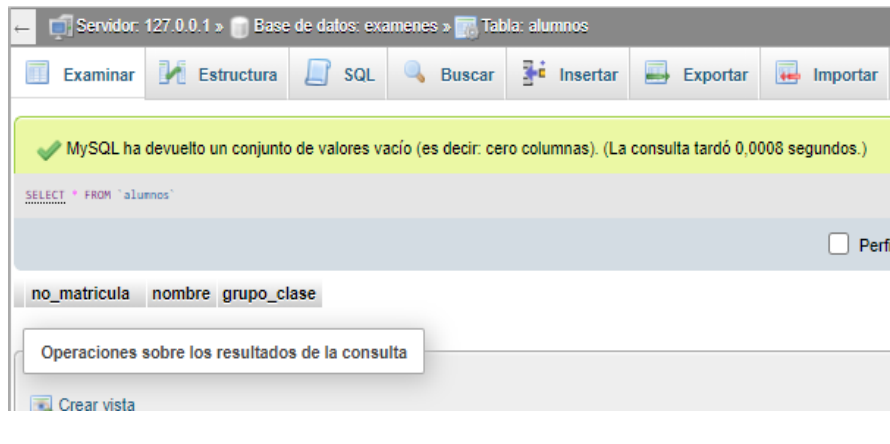


Figura 2.

Importando en la tabla "alumnos"

Archivo a importar:

El archivo puede ser comprimido (gzip, bzip2, zip) o descomprimido.
A compressed file's name must end in .[format].[compression]. Example: .sql.zip

Buscar en su ordenador: No se eligió ningún archivo (Máximo: 40MB)

También puede arrastrar un archivo en cualquier página.

Conjunto de caracteres del archivo:

Importación parcial:

Figura 3.

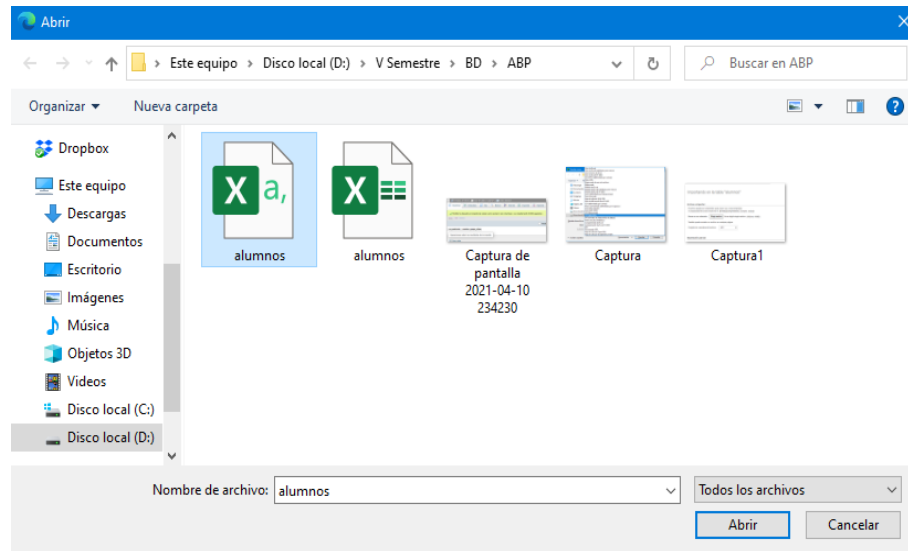


Figura 4.

Opciones específicas al formato:

☐ Actualizar datos cuando las llaves importadas están duplicadas (agregar ON DUPLICATE KEY UPDATE)
Columnas separadas por: Columnas encerradas entre: Caracter de escape de columnas: Líneas terminadas en: Importe este gran número de filas (opcional): nombres de columna:
☐ No abortar si ocurre un error con INSERT

Continuar

Figura 5.

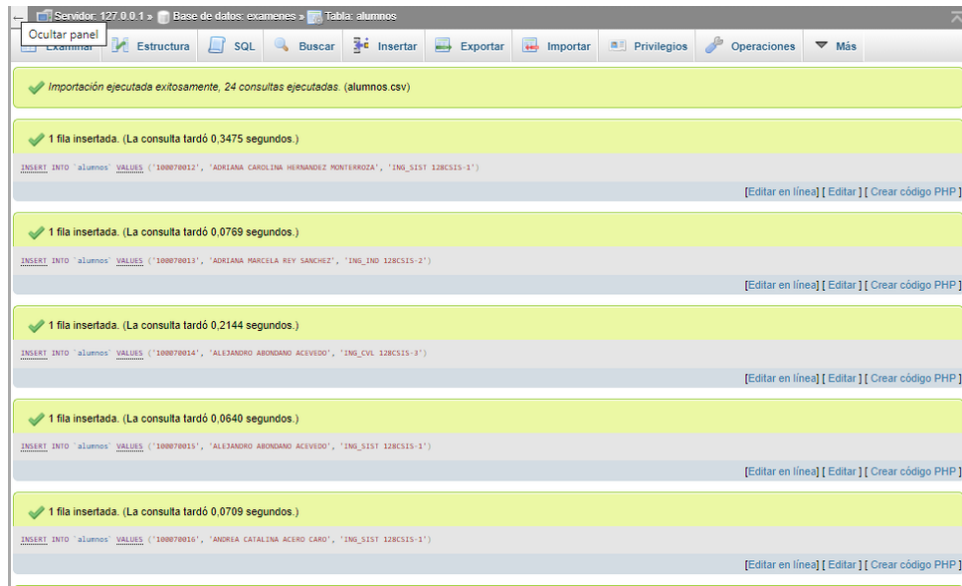


Figura 5.

Servidor: 127.0.0.1 » Base de datos: exámenes » Tabla: alumnos

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Más

			no_matricula	nombre	grupo_clase
<input type="checkbox"/>	Editar	Copiar	Borrar	100070012	ADRIANA CAROLINA HERNANDEZ MONTERROZA ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070013	ADRIANA MARCELA REY SANCHEZ ING_IND 128CSIS-2
<input type="checkbox"/>	Editar	Copiar	Borrar	100070014	ALEJANDRO ABONDANO ACEVEDO ING_CVL 128CSIS-3
<input type="checkbox"/>	Editar	Copiar	Borrar	100070015	ALEJANDRO ABONDANO ACEVEDO ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070016	ANDREA CATALINA ACERO CARO ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070017	ANDREA LILIANA CRUZ GARCIA ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070018	ANDRES FELIPE VILLA MONROY ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070019	ANGELA PATRICIA MAHECHA PIÑEROS ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070020	ANGELICA LISSETH BLANCO CONCHA ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070021	ANGELICA MARIA ROCHA GARCIA ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070022	ANGIE TATIANA FERNÁNDEZ MARTÍNEZ ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070023	TATIANA ANGIE MARTÍNEZ FERNÁNDEZ ING_SIST 128CSIS-1
<input type="checkbox"/>	Editar	Copiar	Borrar	100070024	CAMILO VILLAMIZAR ARISTIZABAL ING_IND 128CSIS-2
<input type="checkbox"/>	Editar	Copiar	Borrar	100070025	CAMILO RODRÍGUEZ BOTERO ING_IND 128CSIS-2
<input type="checkbox"/>	Editar	Copiar	Borrar	100070026	CAMILO ALBERTO CORTÉS MONTEJO ING_IND 128CSIS-2
<input type="checkbox"/>	Editar	Copiar	Borrar	100070027	CAMILO ENRIQUE GÓMEZ RODRÍGUEZ ING_IND 128CSIS-2
<input type="checkbox"/>	Editar	Copiar	Borrar	100070028	CARLOS ANDRÉS POLO CASTELLANOS ING_IND 128CSIS-2
<input type="checkbox"/>	Editar	Copiar	Borrar	100070029	CARLOS DIDIER CASTAÑO CONTRERAS ING_CVL 128CSIS-3

Figura 6.

Sentencias para consultas MySQL

Seleccionamos todo lo que hay en la tabla *alumnos* donde el valor de la columna *nombre* sea todos *nombre* que comienzan con la inicial A.

```
select * from alumnos where nombre like "a%";
```

Seleccionamos la columna *nombre* de la tabla *acceso* donde la columna *codigo* sea igual al valor de: *100070013* y se obtendrá como resultado el *nombre del usuario*.

```
SELECT nombre FROM acceso WHERE codigo = "100070013";
```

Seleccionamos la columna *rol* de la tabla *acceso* donde la columna *codigo* sea igual al valor de *100070013* y se obtendrá como resultado el *rol del usuario*.

```
SELECT rol FROM acceso WHERE codigo = "100070013";
```

Seleccionamos las columnas *codigo* y *password* de la tabla *acceso* donde *codigo* sea igual a: 100070013 y *password* sea igual a: 123456 y se obtendrán sus credenciales de acceso para ser validadas.

```
SELECT codigo, password FROM acceso WHERE codigo = "100070013" && password = "123456" ;
```

Seleccionamos todos los datos que tiene la tabla *alumnos* incluyendo la tabla donde está siendo referenciada (llave foránea) pero al visualizar las tablas sale dos veces la columna ***no_matricula***

```
SELECT * FROM alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula =  
alumnos_controles_escritos.alumnos_no_matricula
```

Seleccionamos todos los datos de la comuna *alumnos* pero un poco más ordenas de la consulta mostrada anteriormente

```
SELECT no_matricula,nombre,grupo_clase,fecha,calificacion,controles_escritos_no_control FROM alumnos INNER JOIN  
alumnos_controles_escritos ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula
```

Actualizamos en la tabla *alumnos_controles_escritos* la columna *calificacion* con el valor de: 4.5 donde la columna *alumnos_no_matricula* sea igual al valor de: 100070013 para editar la antigua calificación del estudiante.

```
UPDATE alumnos_controles_escritos SET calificacion= "4.5" WHERE alumnos_no_matricula= "100070013";
```

Seleccionamos *nombre* y *apellido* de la tabla *profesores* y esto nos regresa el nombre y apellido de todos los profesores registrados en la tabla profesores.

```
SELECT nombre,apellido FROM profesores;
```

Seleccionamos el nombre y apellido, utilizando AS le podemos cambiar por cualquier otro parámetro donde no afectará esos cambios en la base de datos solo se mostrará el cambio cuando se ejecuta la consulta

```
SELECT nombre AS name ,apellido AS surname FROM profesores;
```

Seleccionamos todos los datos distintos de la columna *grupo_clase* de la tabla *alumnos*.

```
SELECT DISTINCT grupo_clase FROM alumnos;
```

Selecciona todos los datos de la tabla *acceso* y nos ordena los nombre de forma ascendente (A-Z)

```
SELECT * FROM acceso ORDER BY nombre ASC
```

Seleccionamos todos los datos de la tabla *acceso* y nos ordena los nombres de forma descendente (Z-A)

```
SELECT * FROM acceso ORDER BY nombre DESC
```

Seleccionamos todos los campos de la tabla *alumnos* y agrupamos a todos los estudiantes que pertenezcan al grupo de clase *ING_SIST 128CSIS-1*


```
SELECT no_matricula,nombre,grupo_clase,fecha,calificacion,controles_escritos_no_control from alumnos INNER JOIN
alumnos_controles_escritos ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula WHERE
grupo_clase ='ING_SIST 128CSIS-1'
```

Seleccionamos todo de la tabla acceso donde el valor de la columna *nombre* sea todos los nombres que inician con la letra *a* y el rol sea *alumno*.

```
SELECT * FROM acceso WHERE nombre LIKE "a%" AND rol = "alumno"
```

Seleccionamos todo de la tabla acceso donde el valor de la columna *nombre* sea todos los nombres que inician con la letra *Y* y el rol sea *profesor*.

```
SELECT * FROM acceso WHERE nombre LIKE "y%" AND rol = "profesor"
```

Seleccionamos la columna *nombre* de la tabla *alumnos* y lo unimos con lo que hay en la tabla *nombre* de la tabla *profesores*.

```
SELECT nombre FROM alumnos UNION ALL SELECT nombre FROM profesores
```

Seleccionamos todas las calificaciones de los alumnos y con la función SUM sumamos todas las calificaciones que se encuentran en la columna calificación

```
SELECT SUM(calificacion) FROM alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula =
```

```
alumnos_controles_escritos.alumnos_no_matricula
```

Seleccionamos todas las calificaciones de los alumnos y con la función AVG sacamos el promedio de las calificaciones de los alumnos y con la función ROUND redondeamos el resultado del promedio

```
SELECT ROUND(AVG(calificacion),2) FROM alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula
```

Seleccionamos el nombre y con la función MAX obtenemos la calificación máxima que un alumno obtuvo

```
SELECT nombre, MAX(calificacion) FROM alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula
```

Seleccionamos todos los datos y con la función COUNT sabemos cuantos datos están registrados en la base de datos

```
SELECT COUNT(codigo) FROM acceso
```

Seleccionamos todos los datos del alumnos y con HAVING sacamos todos los alumnos que tenga una calificación superior a 3.20

```
SELECT no_matricula,nombre,grupo_clase,fecha,calificacion,controles_escritos_no_control FROM alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula HAVING calificacion > 3.20
```

Seleccionamos la columna que queremos que salga todas en mayúscula con la función UPPER de la tabla acceso

```
SELECT UPPER(nombre) from acceso
```

Seleccionamos la columna que queremos que salga todas en minúsculas con la función LOWER de la tabla acceso

```
SELECT LOWER(nombre) from acceso
```

Seleccionamos las columnas grupo_clase y calificación con un alias de PROM de la tabla alumnos donde estamos sacando el promedio de todo los alumnos que pertenecen a una carrera diferente con la función group by

```
SELECT grupo_clase, round(avg(calificacion),2) AS PROM FROM alumnos INNER JOIN alumnos_controles_escritos
ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula group by grupo_clase;
```

Seleccionamos los campos que queremos mostrar de la tabla alumnos que también está relacionada con la tabla alumnoscontrolesescritos, agrupamos todos los datos de los alumnos con su promedio y buscamos por carrera.

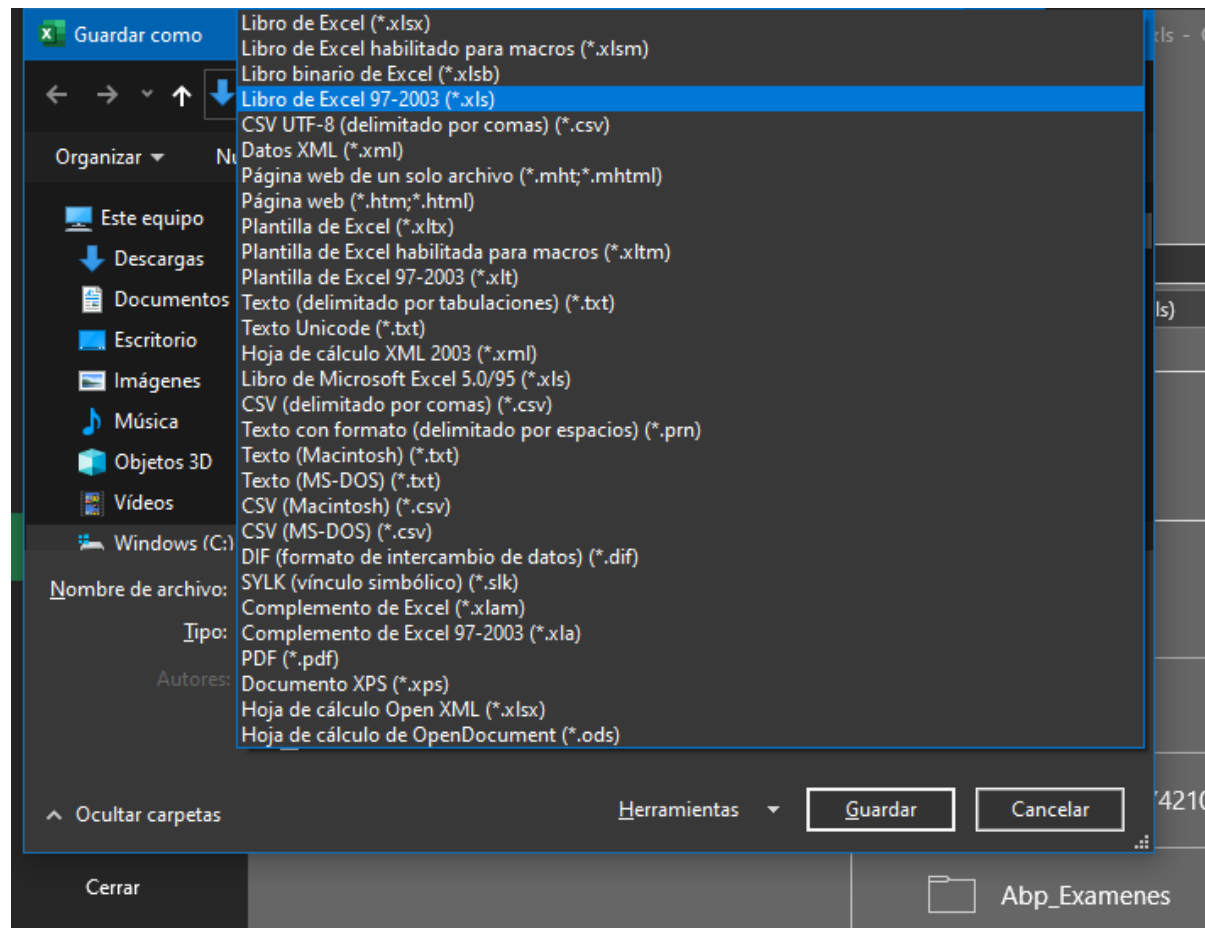
```
select alumnos_no_matricula,nombre, grupo_clase, round(avg(calificacion),2)AS prom from vistaalumnos inner join
vistaalumnoscontrolesescritos
on vistaalumnos.no_matricula = vistaalumnoscontrolesescritos.alumnos_no_matricula
group by alumnos_no_matricula having grupo_clase = 'ING_SIST 128CSIS-1'
```

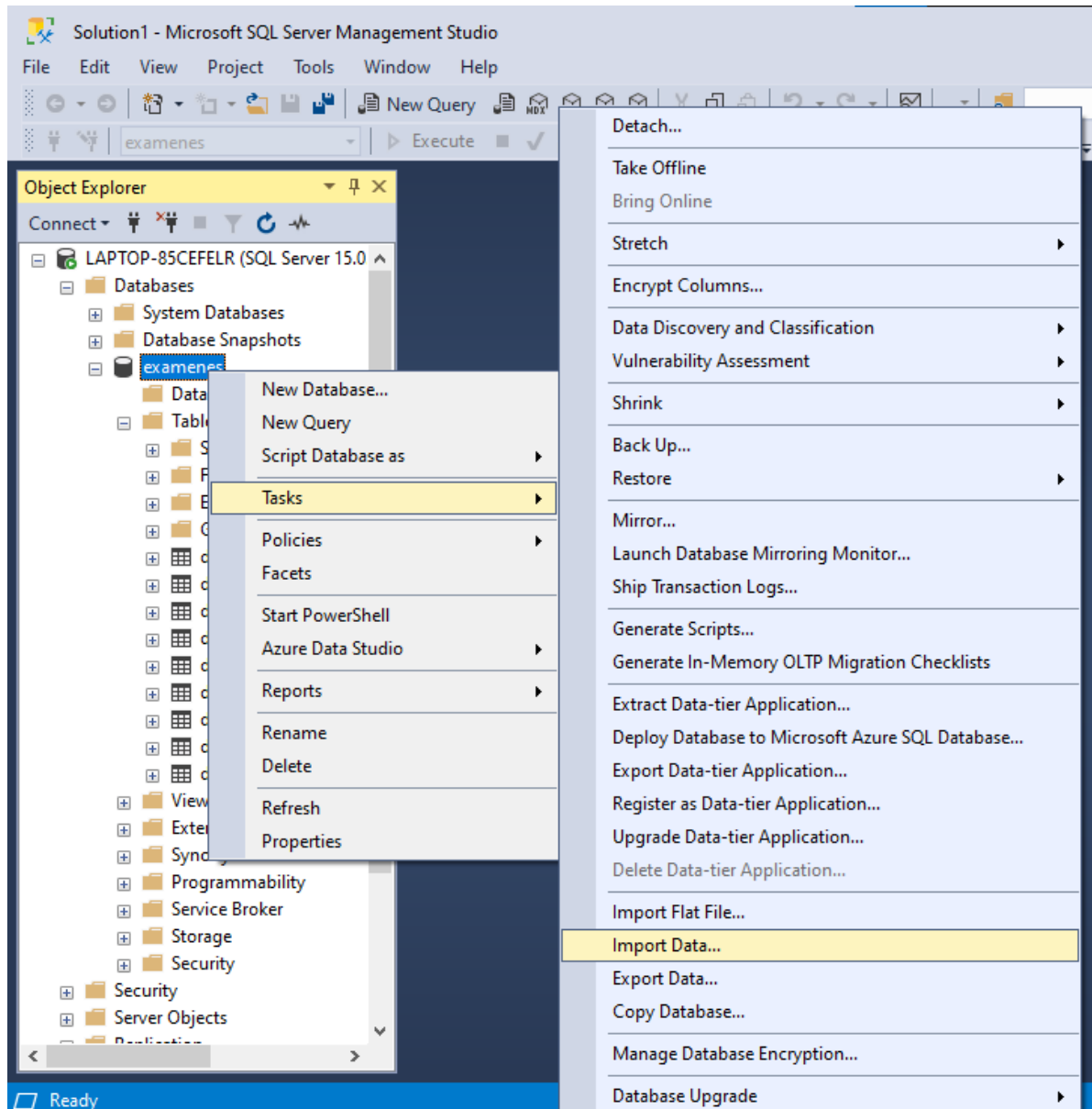
Exportar un archivo Excel a una base de datos SQL Server

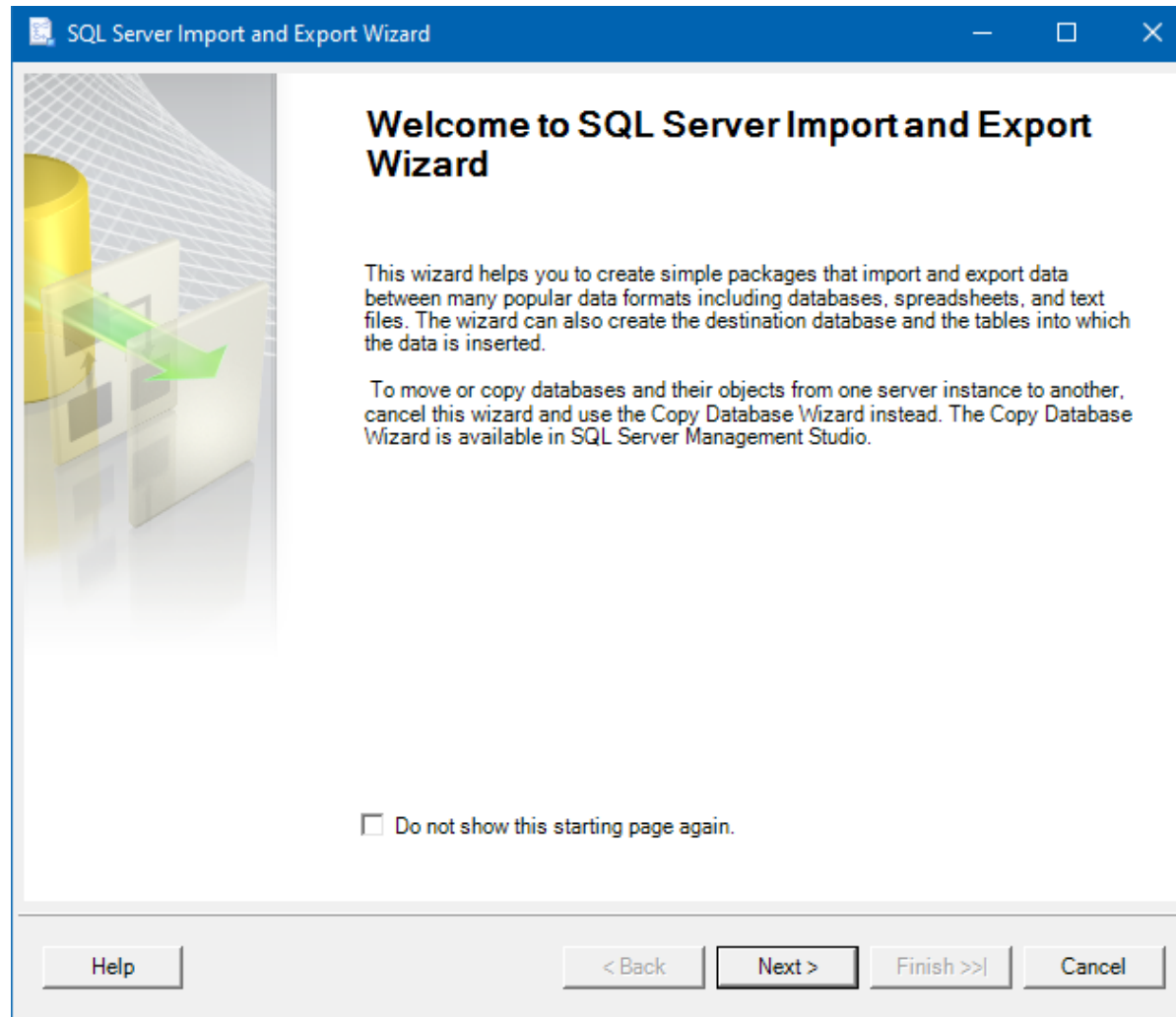
Para exportar un Archivo Excel a una base de datos SQL server, luego de hacer la respectiva limpieza y estructuración de los datos realizamos los siguientes pasos:

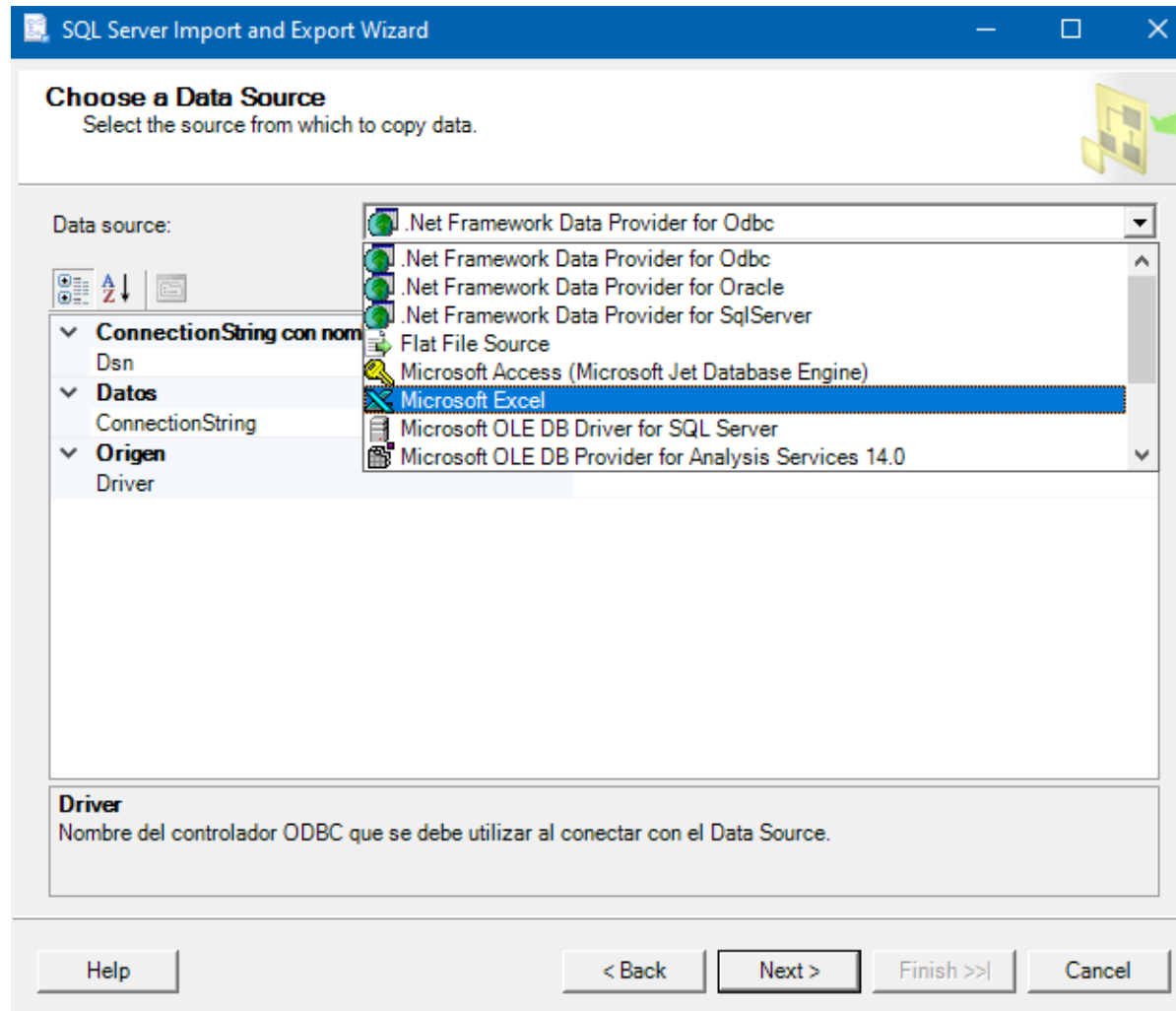
1. Guardamos el archivo excel con el formato *Libro de Excel 97-2003 (*.xls)* como en la *figura 1*.
2. En Microsoft SQL server management studio buscamos la base de datos donde vamos a importar los datos y le damos click derecho -> *Tasks -> Import Data...* como en la *figura número 2*.
3. Abrirá el asistente de importación y exportación le damos *next*.
4. Seleccionamos en *data source: Microsoft Excel* que es de donde exportamos los datos como en la *figura 4*.
5. En Excel file patch, buscaremos el archivo que se guardó en el *paso número 1* y Excel versión usaremos *Microsoft Excel 97-2003* como se ve en la *figura número 5* y damos *next*.
6. En *destination* seleccionamos: *SQL server native client 11.0* como en la *figura 6*.
7. En *database* seleccionamos la base de datos donde se van a importar los datos como en la *figura número 7* y le damos *next*.
8. Seleccionamos *Copy data from one or more tables or views* y damos *next* como en la *figura 8*.

9. Seleccionamos la tabla donde se van a importar nuestros datos como en la *figura número 9* y le damos en next.
10. Le damos next.
11. Se deja marcada la opción *Run immediately* y le damos en finish.
12. Para comprobar que los datos se guardaron se hace un query `SELECT * FROM profesores;` y este nos devuelve todos los datos que hay en la tabla profesores que serían los que acabamos de importar.









The screenshot shows the 'SQL Server Import and Export Wizard' window, specifically the 'Choose a Data Source' step. The window has a blue title bar with the text 'SQL Server Import and Export Wizard'. Below the title bar, the step is titled 'Choose a Data Source' with the instruction 'Select the source from which to copy data.' and a small icon of a folder with a green arrow. The 'Data source:' dropdown menu is set to 'Microsoft Excel'. Below this, the 'Excel connection settings' section contains the following fields: 'Excel file path:' with a text box containing 'C:\Users\boris\Downloads\profesores (1).xls' and a 'Browse...' button; 'Excel version:' with a dropdown menu set to 'Microsoft Excel 97-2003'; and a checked checkbox labeled 'First row has column names'. At the bottom of the window, there are five buttons: 'Help', '< Back', 'Next >', 'Finish >>', and 'Cancel'.

SQL Server Import and Export Wizard

Choose a Data Source
Select the source from which to copy data.

Data source: Microsoft Excel

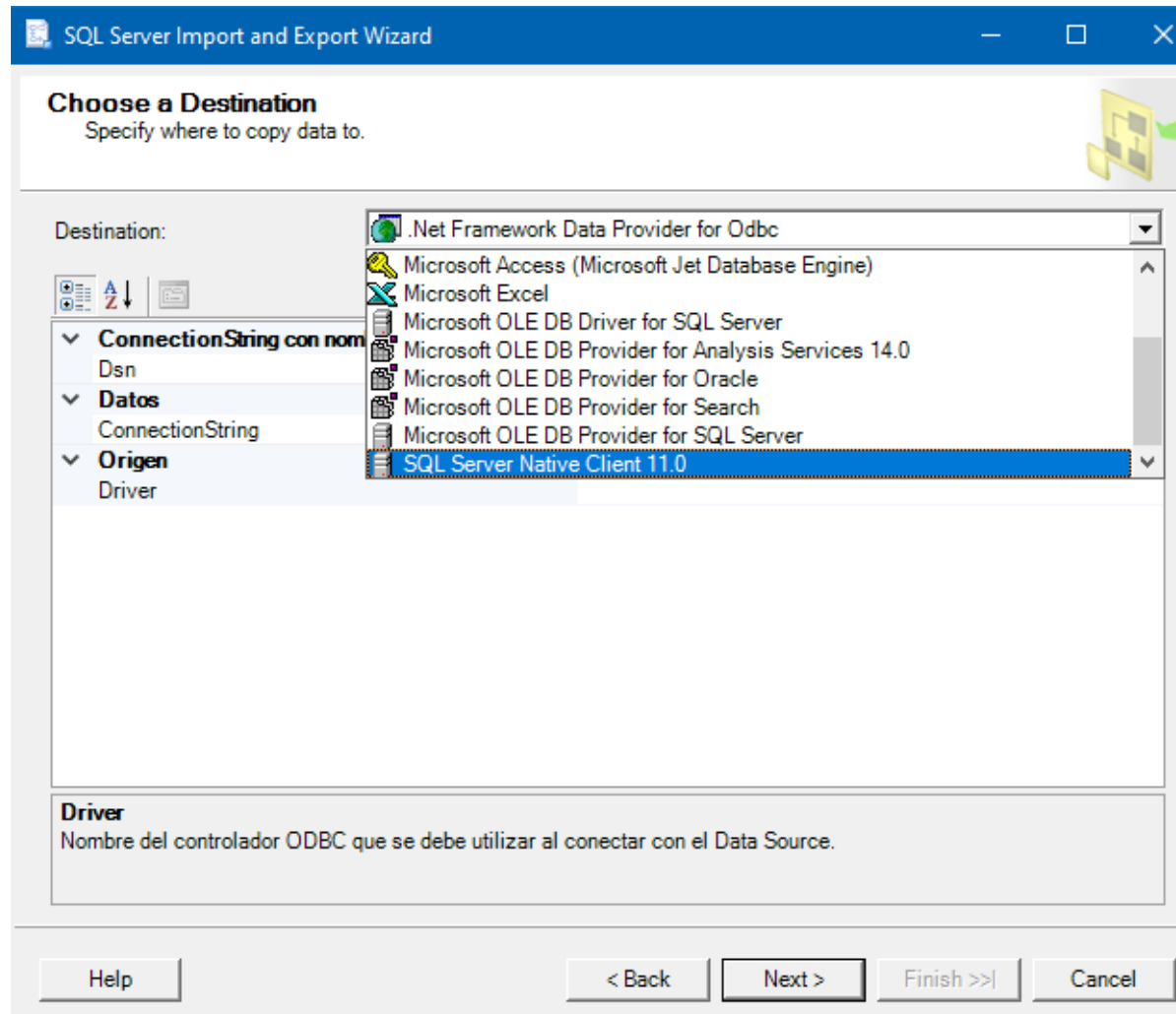
Excel connection settings

Excel file path:
C:\Users\boris\Downloads\profesores (1).xls Browse...

Excel version:
Microsoft Excel 97-2003

☒ First row has column names

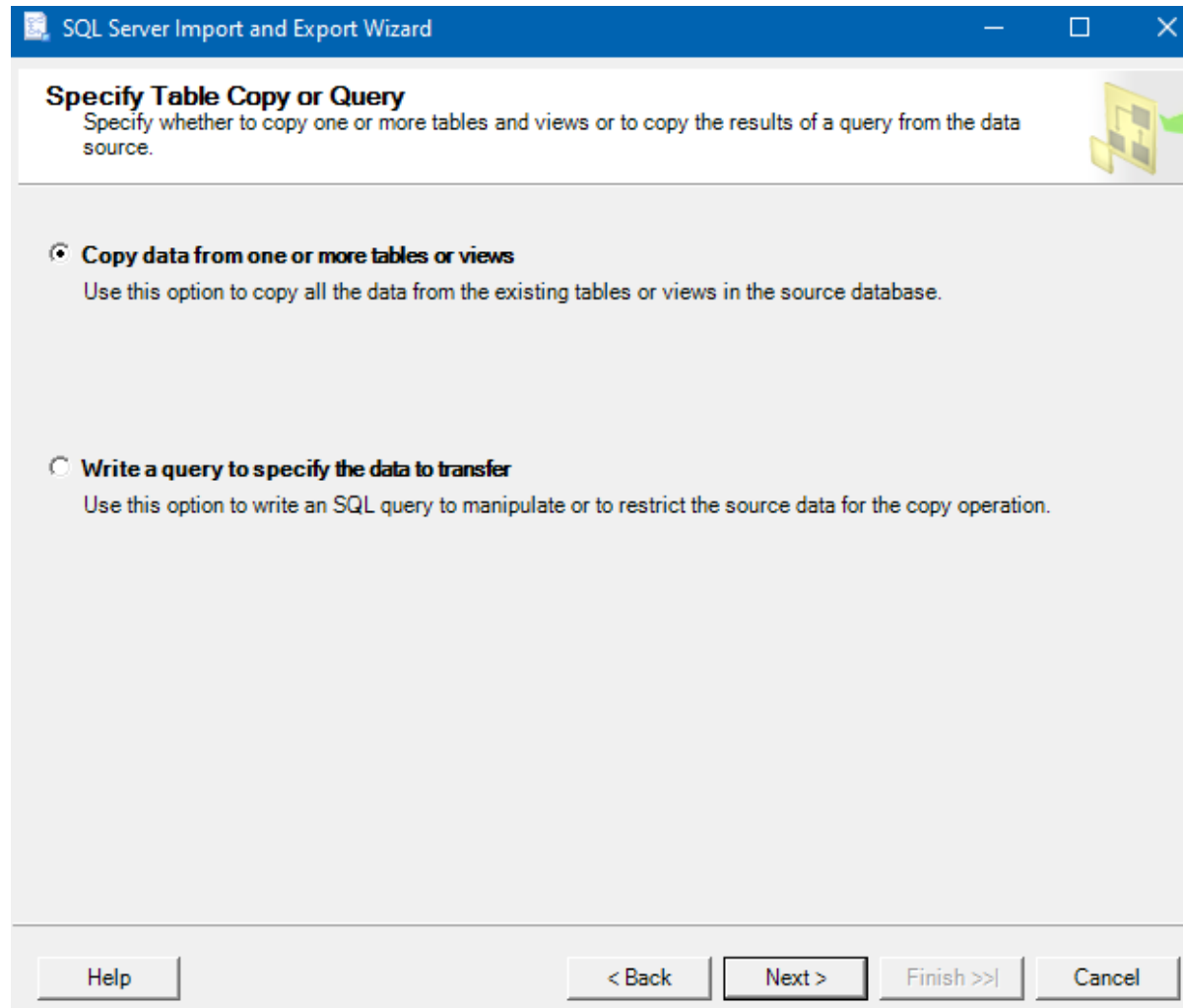
Help < Back Next > Finish >> Cancel

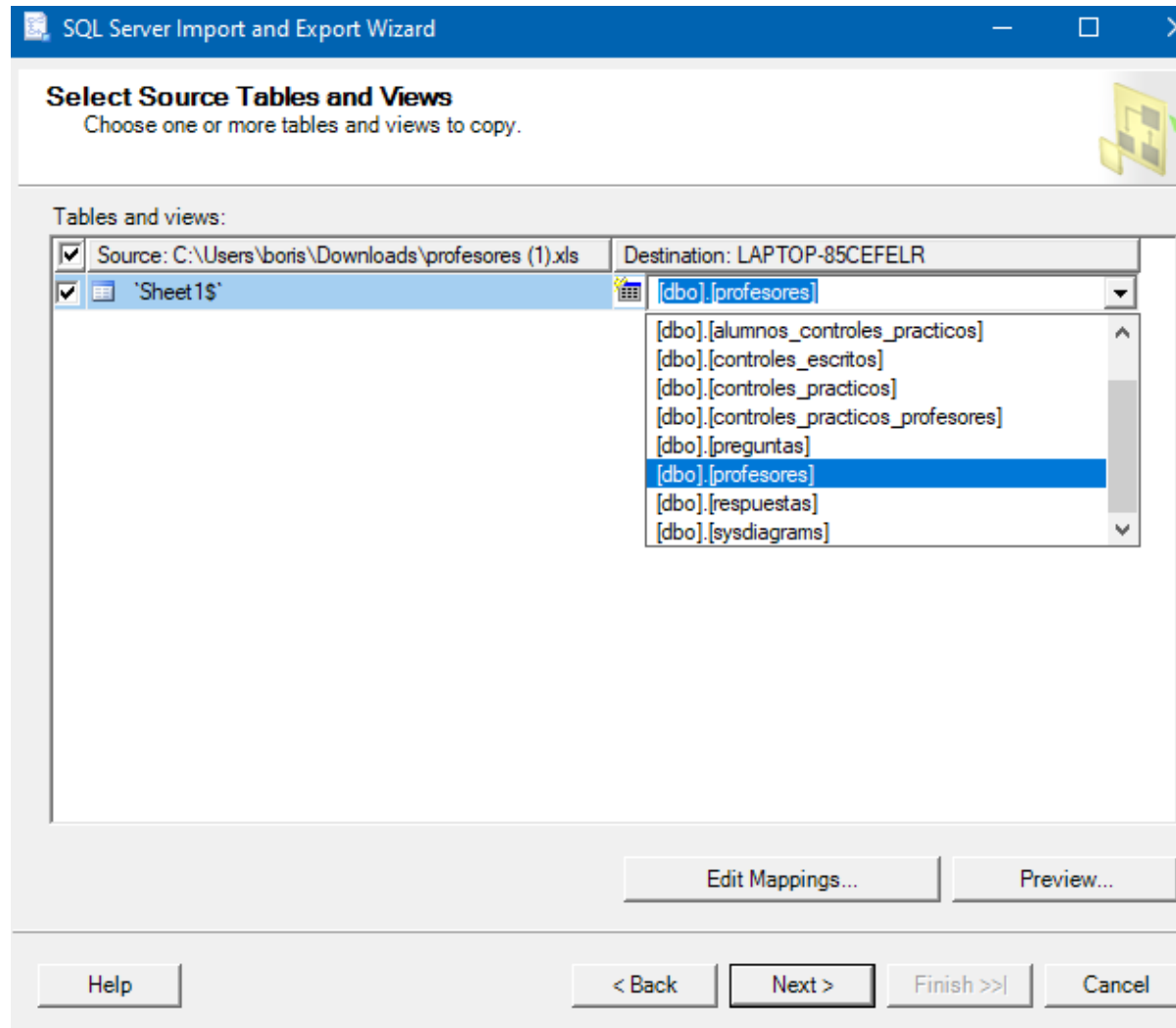


The screenshot shows the 'SQL Server Import and Export Wizard' window, specifically the 'Choose a Destination' step. The window has a blue title bar with the text 'SQL Server Import and Export Wizard'. Below the title bar, the main area is titled 'Choose a Destination' with the subtitle 'Specify where to copy data to.' and a yellow folder icon with a green arrow. The form contains the following fields and controls:

- Destination:** A dropdown menu showing 'SQL Server Native Client 11.0'.
- Server name:** A dropdown menu showing 'LAPTOP-85CEFELR'.
- Authentication:** A section with two radio buttons: 'Use Windows Authentication' (selected) and 'Use SQL Server Authentication'. Below these are text boxes for 'User name:' and 'Password:'.
- Database:** A dropdown menu showing 'examenes', with 'Refresh' and 'New...' buttons to its right.

At the bottom of the window, there is a row of buttons: 'Help', '< Back', 'Next >', 'Finish >>', and 'Cancel'.





SQL Server Import and Export Wizard

Review Data Type Mapping

Select a table to review how its data types map to those in the destination and how it handles conversion issues.

Table:

Source	Destination
'Sheet1\$'	[dbo].[profesores]

Data type mapping:

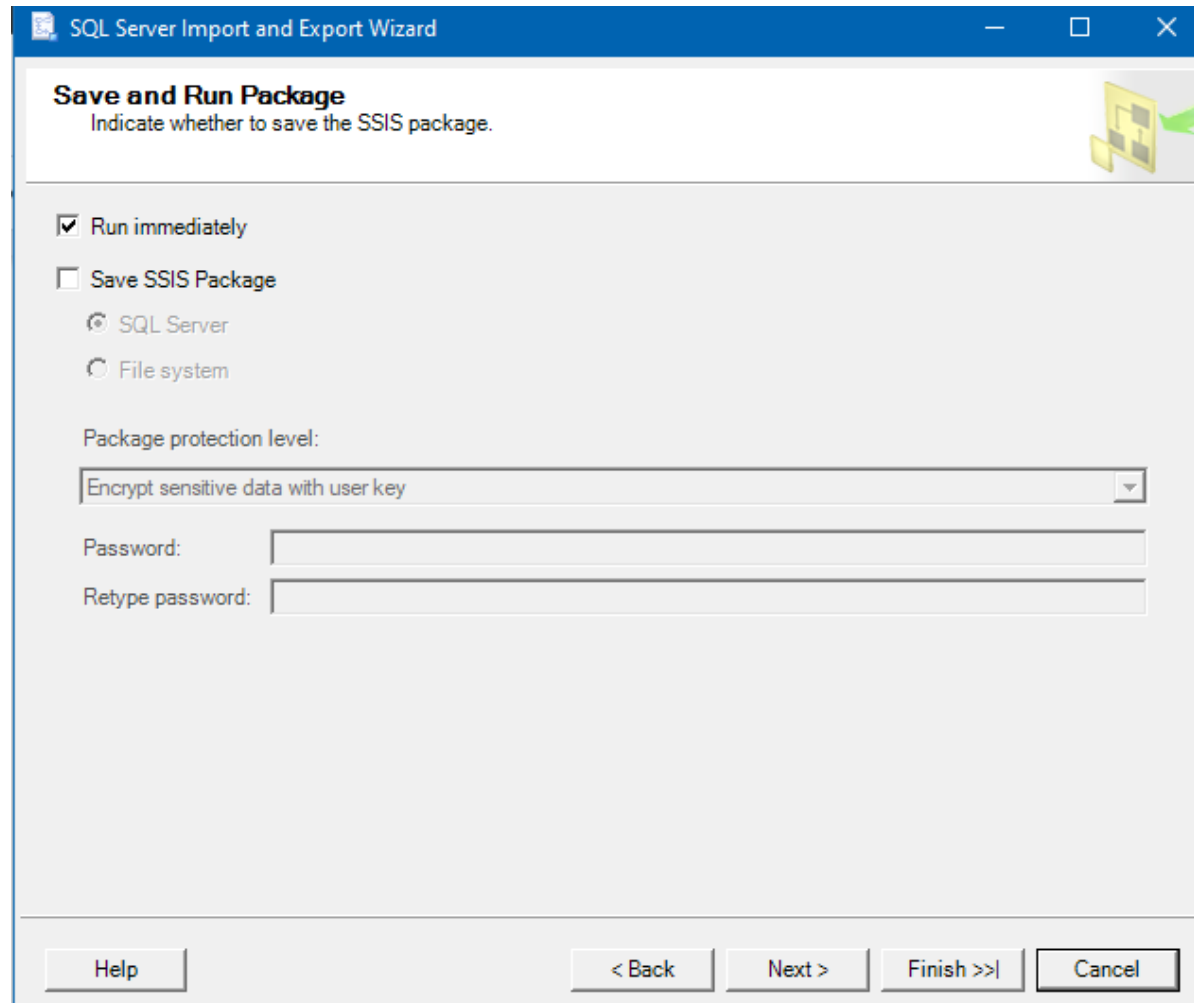
	Source Column	Source Type	Destination Col...	Destination Type	Convert	On Error	On Truncati..
	dni	VarChar	dni	varchar	<input checked="" type="checkbox"/>	Use Global	Use Global
	nombre	VarChar	nombre	varchar	<input checked="" type="checkbox"/>	Use Global	Use Global
	apellido	VarChar	apellido	varchar	<input checked="" type="checkbox"/>	Use Global	Use Global

To view conversion details, double-click the row that contains the column source type to be converted.

On Error (global) Fail

On Truncation (global) Fail

Help < Back Next > Finish >> Cancel



The screenshot shows the 'Save and Run Package' step of the SQL Server Import and Export Wizard. The window title is 'SQL Server Import and Export Wizard'. The main heading is 'Save and Run Package' with the instruction 'Indicate whether to save the SSIS package.' and a yellow cube icon with a green arrow. The options are as follows:

- ☒ Run immediately
- ☐ Save SSIS Package
 - ☒ SQL Server
 - ☐ File system

Package protection level:
Encrypt sensitive data with user key (dropdown menu)

Password: [text box]
Retype password: [text box]

Buttons at the bottom: Help, < Back, Next >, Finish >>, Cancel.

SELECT * FROM profesores;

100 %

Results Messages

	dni	nombre	apellido
1	UNI-1020	MÓNICA ALEXANDRA	CAMACHO AMAYA
2	UNI-1021	JOSE GUILLERMO	MARIN ZUBIETA
3	UNI-1022	HUGO ANDRÉS	CAMARGO VARGAS
4	UNI-1023	INGRID ROCIO	GUERRERO PENAGOS
5	UNI-1024	IVÁN DAVID	CORAL BURBANO
6	UNI-1025	IVONNE JOULIETTE	BARRERA LOPEZ
7	UNI-1026	JENNY FERNANDA	SÁNCHEZ ARENAS
8	UNI-1027	JENNY VIVIANA	MONCALEANO PRECIADO
9	UNI-1028	JORGE ESTEBAN	REY BOTERO
10	UNI-1029	JORGE MARIO	OROZCO DUSSÁN
11	UNI-1030	MÓNICA NATALIA	CAMARGO MENDOZA

Query executed successfully.

Sentencias para consultas para SQL Server

Seleccionamos todos los datos de la tabla *alumnos* donde el nombre empiece con la letra A.

```
SELECT * FROM alumnos WHERE nombre like 'A%';
```

Seleccionamos el *nombre* y *rol* de la tabla *acceso* donde el código sea igual a: 100070013 y se obtendrá el nombre y rol de la persona identificada con el código anterior en caso de existir.

```
SELECT nombre,rol FROM acceso WHERE codigo = '100070013';
```

Actualizamos en la tabla *alumnos_controles_escritos* la columna *calificacion* con el valor de: 4.5 donde la columna *alumnos_no_matricula* sea igual al valor de: 100070013 para editar la antigua calificación del estudiante.

```
UPDATE alumnos_controles_escritos SET calificacion= 4.5 WHERE alumnos_no_matricula= 100070013;
```

Seleccionamos la calificación y sacamos el promedio de calificaciones que están en la tabla *alumnos*.

```
SELECT AVG(calificacion) FROM alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula
```

Seleccionamos de la tabla *alumnos* relacionada con la tabla *alumnos_controles_escritos* todos los usuarios cuyo promedio sea mayor o igual a 3 y su grupo de clase sea 'ING_IND 128CSIS-2'

```
select no_matricula AS codigo,nombre AS name,grupo_clase AS Carrera,calificacion AS nota from alumnos INNER JOIN
alumnos_controles_escritos ON alumnos.no_matricula
= alumnos_controles_escritos.alumnos_no_matricula WHERE calificacion >=3 AND grupo_clase = 'ING_IND 128CSIS-2'
```

Seleccionamos la tabla alumnos y nos mostrará los datos distintos de la columna *grupo_clase*

```
SELECT DISTINCT grupo_clase FROM alumnos;
```

Seleccionamos la calificación y sacamos el promedio de calificaciones que están en la tabla *alumnos* solo para los estudiantes pertenecientes al grupo de clase: *ING_SIST 128CSIS-1*.

```
SELECT AVG(calificacion) AS Prom_carrera FROM alumnos INNER JOIN alumnos_controles_escritos ON
alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula WHERE grupo_clase = 'ING_SIST 128CSIS-1'
```

Seleccionamos la calificación y sacamos el promedio de calificaciones que están en la tabla *alumnos*.

```
SELECT AVG(calificacion) FROM alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula =
alumnos_controles_escritos.alumnos_no_matricula
```

Exportar un archivo Excel a una base de datos PostgreSQL

Para exportar un Archivo Excel a una base de datos PostgreSQL, luego de hacer la respectiva limpieza y estructuración de los datos realizamos los siguientes pasos:

- 1) Guardamos el archivo de excel con la extensión .CSV tal como se muestra en la *figura 1*.
- 2) Buscamos el archivo guardado en el paso anterior y le damos clic derecho y *Abrir con* elegimos bloc de notas, tal como se evidencia en la *figura 2*.
- 3) Una vez en el bloc de notas le damos en *guardar como*, tal como se muestra en la *figura 3*.
- 4) Buscamos la ruta para guardar el archivo y en la codificación se selecciona *UTF-8* y guardamos como se observa en la *figura 4*.
- 5) Usamos la siguiente sentencia para agregar los datos a la tabla en PostgreSQL: `COPY acceso FROM 'D:\acceso.txt' USING DELIMITERS ';' ;` donde le estamos indicando que copie en la tabla acceso lo que se encuentra en la ruta: *D:\acceso.txt* que en este caso serían los datos que guardamos en el paso anterior.
- 6) Ejecutamos un `Select * FROM acceso;` y comprobamos que los datos se han cargado correctamente.

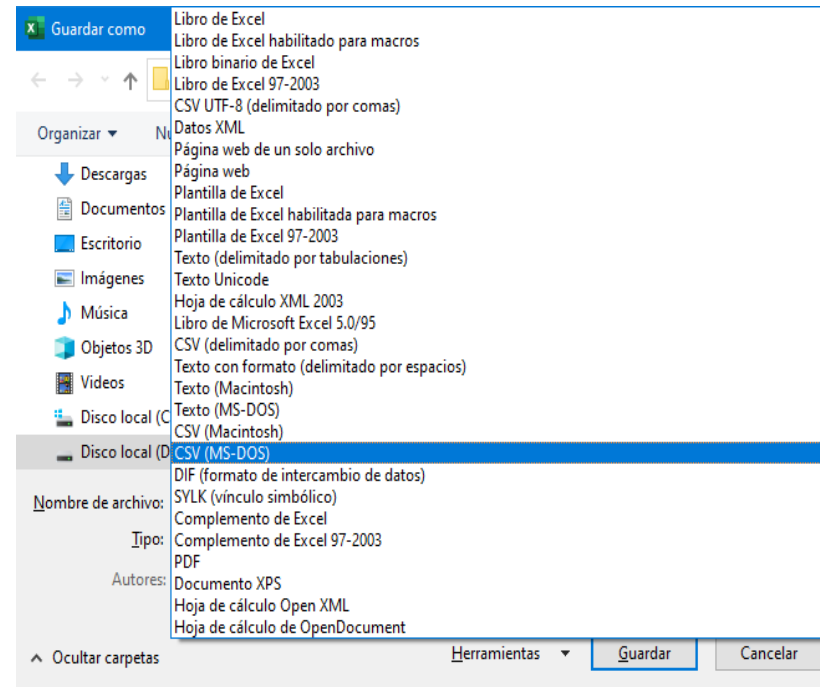


figura 1

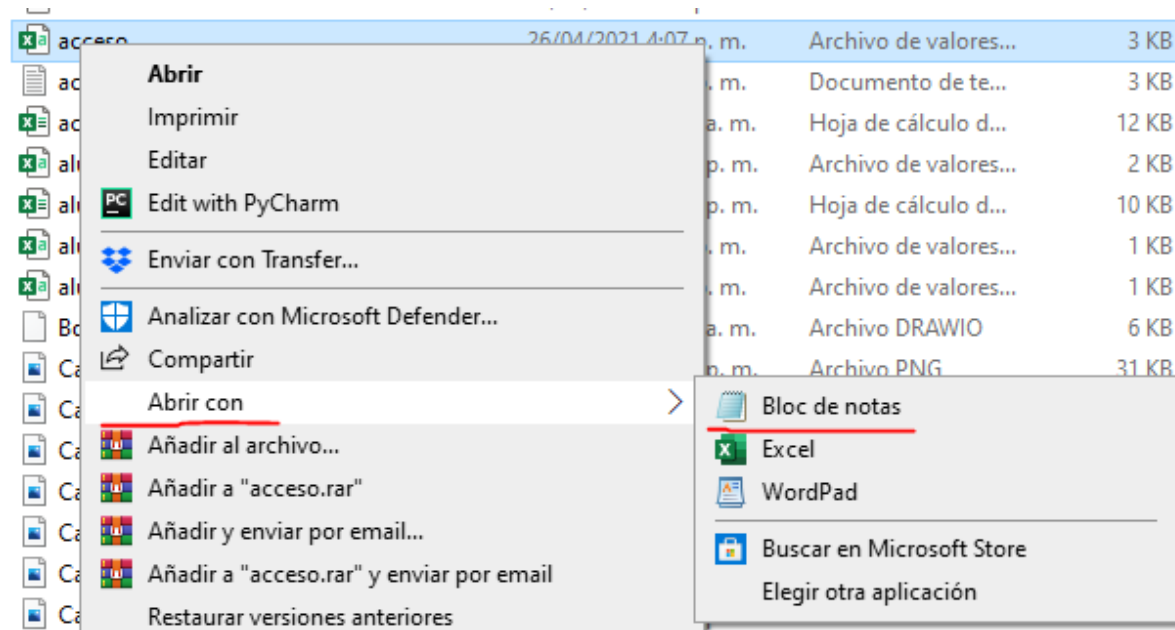


figura 2

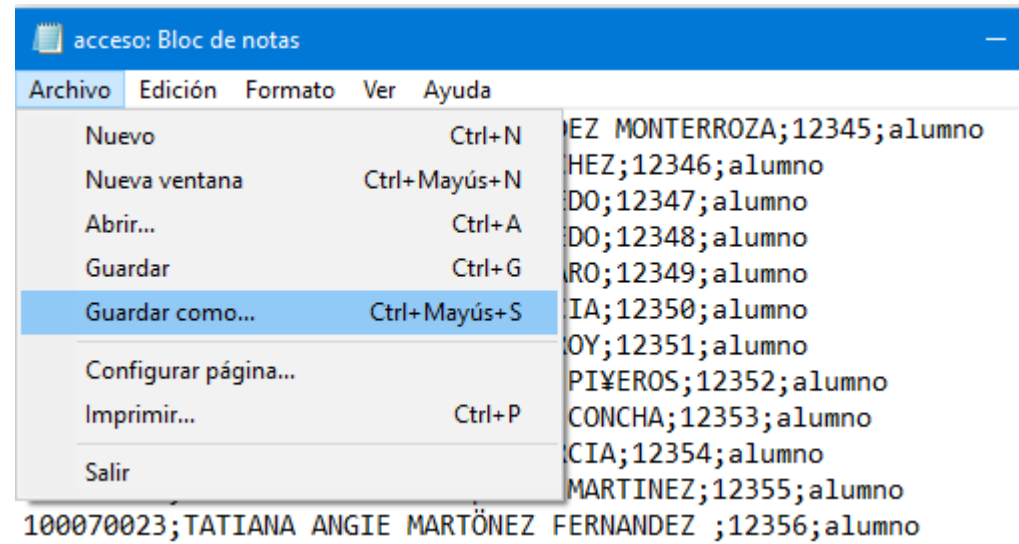


figura 3

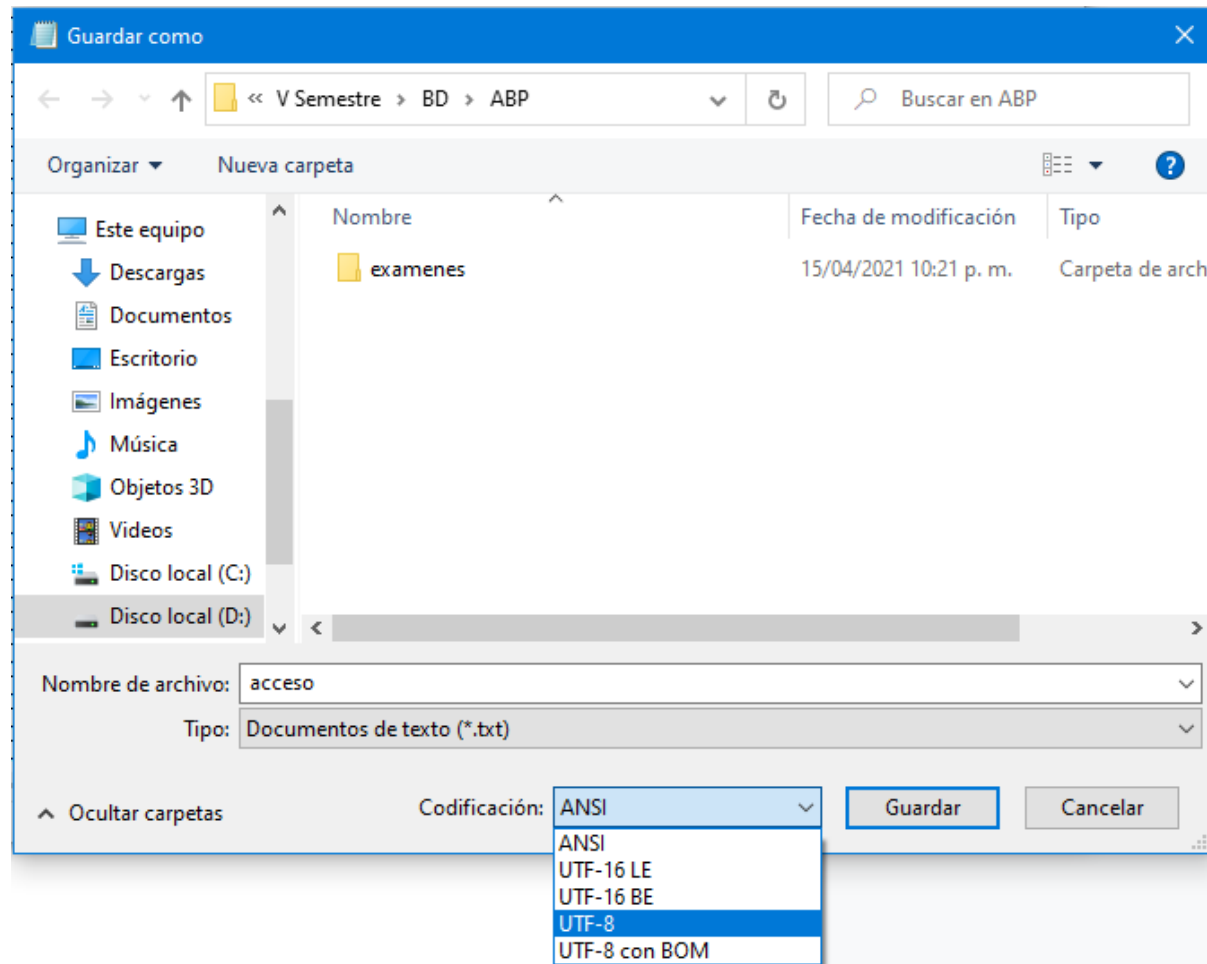


figura 4

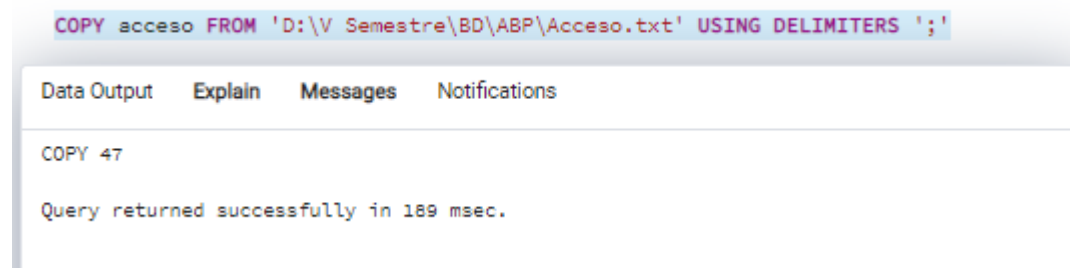


figura 5

Select * from acceso;

Data Output Explain Messages Notifications

	codigo character varying (50)	nombre character varying (50)	password character varying (50)	rol character varying (40)
1	100070012	ADRIANA CAROLINA ...	12345	alumno
2	100070013	ADRIANA MARCELA R...	12346	alumno
3	100070014	ALEJANDRO ABONDA...	12347	alumno
4	100070015	ALEJANDRO ABONDA...	12348	alumno
5	100070016	ANDREA CATALINA A...	12349	alumno
6	100070017	ANDREA LILIANA CR...	12350	alumno
7	100070018	ANDRES FELIPE VILL...	12351	alumno
8	100070019	ANGELA PATRICIA M...	12352	alumno
9	100070020	ANGELICA LISSETH B...	12353	alumno
10	100070021	ANGELICA MARIA RO...	12354	alumno
11	100070022	ANGIE TATIANA FERN...	12355	alumno
12	100070023	TATIANA ANGIE MAR...	12356	alumno
13	100070024	CAMILO VILLAMIZAR ...	12357	alumno
14	100070025	CAMILO RODRÖGUEZ ...	12358	alumno

figura 6.

Sentencias para consultas para PostgreSQL

Seleccionamos todo lo que hay en la tabla *alumnos* donde el valor de la columna *nombre* sea todos *nombre* que comienzan con la letra A.

```
SELECT * FROM alumnos WHERE nombre like 'A%';
```

Seleccionamos el *nombre* y *rol* de la tabla *acceso* donde el código sea igual a: *100070013* y se obtendrá el nombre y rol de la persona identificada con el código anterior en caso de existir.

```
SELECT nombre,rol FROM acceso WHERE codigo = '100070013';
```

Actualizamos en la tabla *alumnos_controles_escritos* la columna *calificacion* con el valor de: 4.5 donde la columna *alumnos_no_matricula* sea igual al valor de: *100070013* para editar la antigua calificación del estudiante.

```
UPDATE alumnos_controles_escritos SET calificacion= 4.5 WHERE alumnos_no_matricula= 100070013;
```

Seleccionamos todos los campos de la tabla *alumnos* y agrupamos a todos los estudiantes que pertenezcan al grupo de clase *ING_SIST 128CSIS-1*.

```
SELECT no_matricula,nombre,grupo_clase,fecha,calificacion,controles_escritos_no_control from alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula WHERE
```

```
grupo_clase = 'ING_SIST 128CSIS-1'
```

Seleccionamos la columna que queremos que salga todas en minúsculas con la función LOWER de la tabla acceso

```
SELECT LOWER(nombre) from acceso
```

Seleccionamos la calificación y sacamos el promedio de calificaciones que están en la tabla *alumnos*.

```
SELECT AVG(calificacion) FROM alumnos INNER JOIN alumnos_controles_escritos ON alumnos.no_matricula =  
alumnos_controles_escritos.alumnos_no_matricula
```

Seleccionamos la tabla *alumnos* y nos mostrará los datos distintos de la columna *grupo_clase*

```
SELECT DISTINCT grupo_clase FROM alumnos;
```

Seleccionamos de la tabla *alumnos* relacionada con la tabla *alumnos_controles_escritos* todos los usuarios cuyo promedio sea mayor o igual a 3 y su grupo de clase sea 'ING_IND 128CSIS-2'

```
select no_matricula AS codigo,nombre AS name,grupo_clase AS Carrera,calificacion AS nota from alumnos INNER JOIN  
alumnos_controles_escritos ON alumnos.no_matricula  
= alumnos_controles_escritos.alumnos_no_matricula WHERE calificacion >=3 AND grupo_clase = 'ING_IND 128CSIS-2'
```

Seleccionamos la calificación y sacamos el promedio de calificaciones que están en la tabla *alumnos* solo para los estudiantes pertenecientes al grupo de clase: *ING_SIST 128CSIS-1*.

```
SELECT AVG(calificacion) AS Prom_carrera FROM alumnos INNER JOIN alumnos_controles_escritos ON  
alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula WHERE grupo_clase = 'ING_SIST 128CSIS-1'
```

Exportar un archivo Excel a una base de datos Oracle

Para exportar un Archivo Excel a una base de datos SQL server, luego de hacer la respectiva limpieza y estructuración de los datos realizamos los siguientes pasos:

1. Guardamos el archivo excel con el formato CSV.
2. *Seleccionamos la tabla donde se va a importar los datos, click derecho y seleccionamos importar datos como se muestra en la figura 1.*
3. *Luego de haber presionado click en importar datos le damos examinar para ubicar el archivo donde se ha guardado con la extensión CSV. figura 2.*
4. *Seleccionamos el archivo donde están todos los datos que queremos guardar. figura 3.*
5. *Una vez de haber seleccionado el archivo nos aparecerá la siguiente ventana al cual procedemos a darle siguiente. figura 4*
6. *En la siguiente imagen procedemos a darle siguiente. figura 5.*
7. *Después de haber realizado el paso anterior, en la ventana presente presionamos terminar. figura 6*
8. *Una vez de haber importado y terminado el proceso vemos las consultas que se hicieron correctamente. figura 7*
9. *Para visualizar los datos demostrando que los datos fueron insertados correctamente ingresamos `SELECT * FROM aLumnos;` y visualizamos todos los datos importados.*

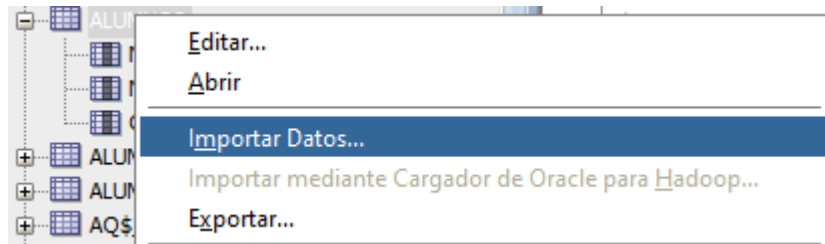


Figura 1.



Figura 2.

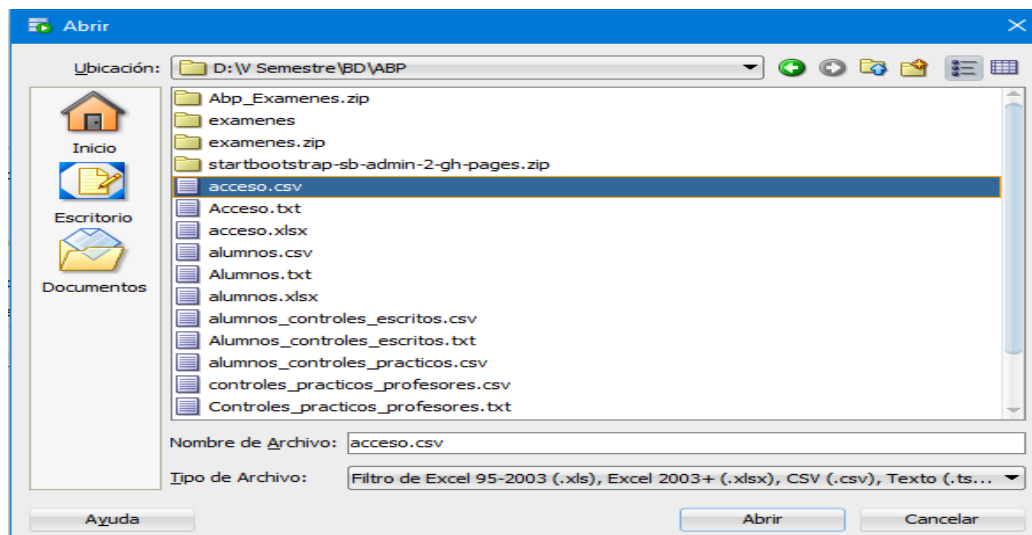


Figura 3.

Método de Importación

Especifique el método para importar los datos. En el caso del método de tabla externa en área temporal, se creará una tabla externa como tabla intermedia para importar la tabla de destino. Para otros métodos de importación, los datos se importan directamente a la tabla.

Método de Importación:

☐ Enviar Crear Script a Hoja de Trabajo de SQL

Nombre de la Tabla:

☐ Límite de Importación de Filas:

Contenido del Archivo

100070013;ADRIANA MARCELA REY SANCHEZ;ING_IND 128CSIS-2
100070014;ALEJANDRO ABONDANO ACEVEDO;ING_CVL 128CSIS-3
100070015;ALEJANDRO ABONDANO ACEVEDO;ING_SIST 128CSIS-1
100070016;ANDREA CATALINA ACERO CARO;ING_SIST 128CSIS-1
100070017;ANDREA LILIANA CRUZ GARCIA;ING_SIST 128CSIS-1
100070018;ANDRES FELIPE VILLA MONROY;ING_SIST 128CSIS-1
100070019;ANGELA PATRICIA MAHECHA PIVEROS;ING_SIST 128CSIS-1
100070020;ANGELICA LISSETH BLANCO CONCHA;ING_SIST 128CSIS-1
100070021;ANGELICA MARIA ROCHA GARCIA;ING_SIST 128CSIS-1
100070022;ANGIE TATIANA FERNANDEZ MARTÍNEZ;ING_SIST 128CSIS-1
100070023;TATIANA ANGIE MARTÍNEZ FERNANDEZ;ING_SIST 128CSIS-1
100070024;CAMILO VILLAMIZAR ARISTIZABAL;ING_IND 128CSIS-2
100070025;CAMILO RODRÍGUEZ BOTERO;ING_IND 128CSIS-2
100070026;CAMILO ALBERTO CORTÉS MONTEJO;ING_IND 128CSIS-2
100070027;CAMILO ENRIQUE GÓMEZ RODRÍGUEZ;ING_IND 128CSIS-2
100070028;CARLOS ANDRÉS POLO CASTELLANOS;ING_IND 128CSIS-2
100070029;CARLOS DIDIER CASTAÑO CONTRERAS;ING_CVL 128CSIS-3
100070030;CAROL RUCHINA GÓMEZ GIANINI;ING_CVL 128CSIS-3

Ayuda < Atrás **Siguiente >** Terminar Cancelar

Figura 4.

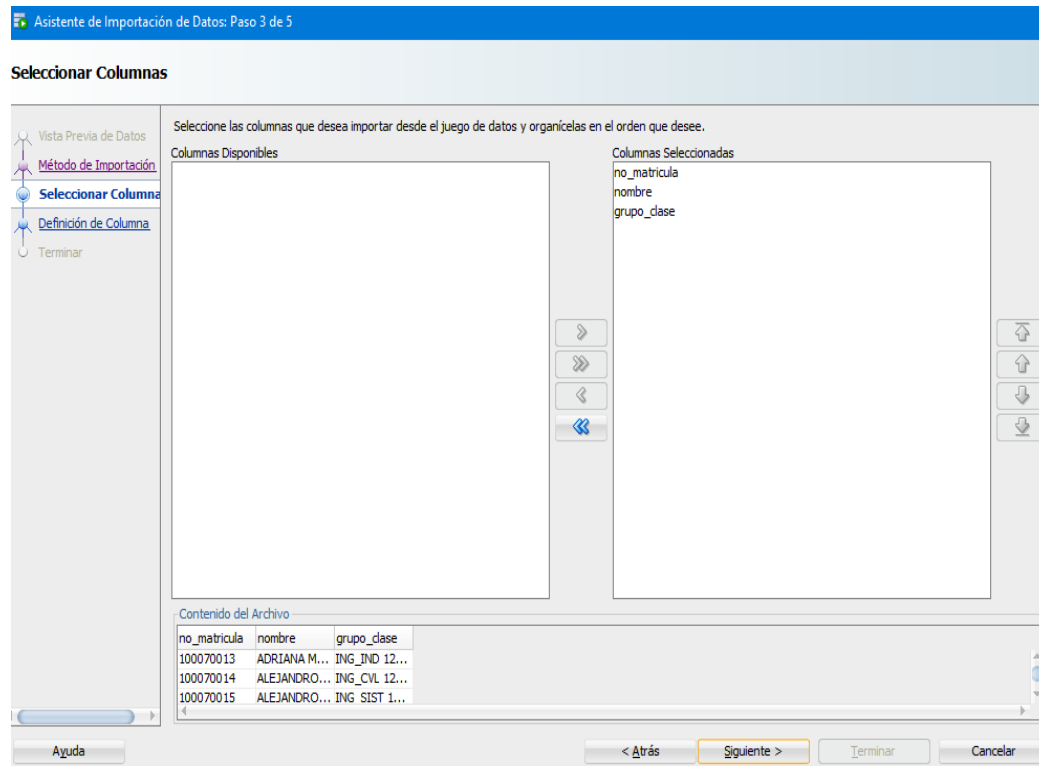


Figura 5.

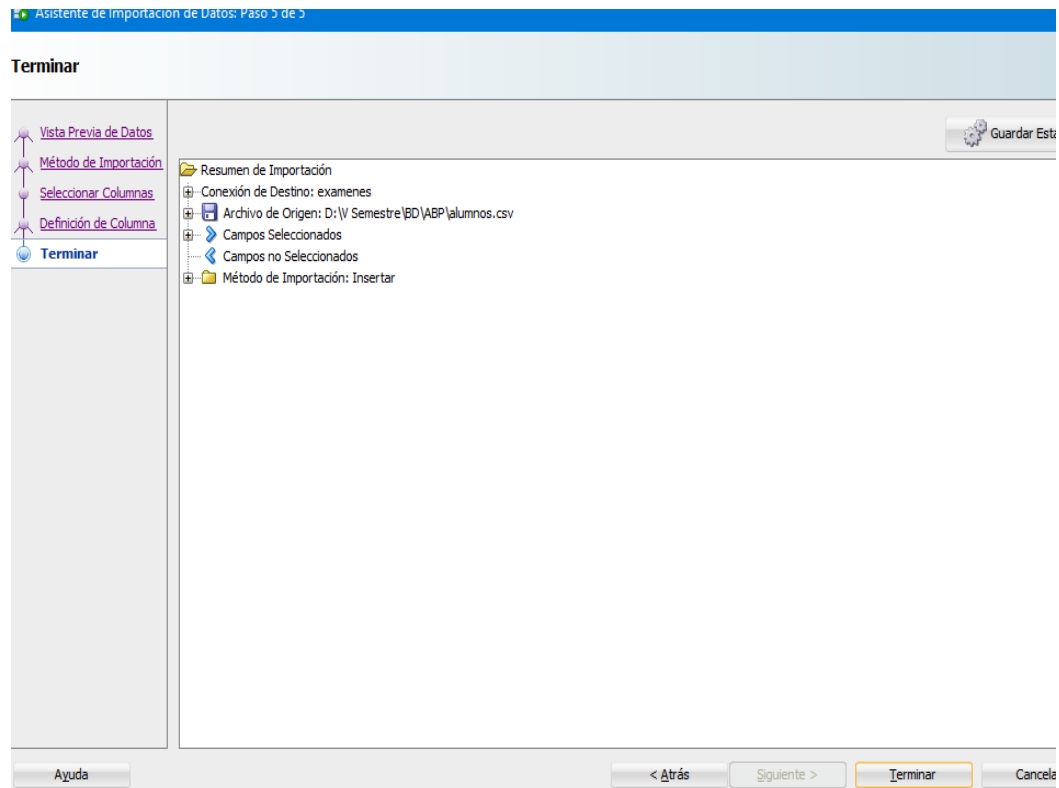


Figura 6.

```

INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070020,'ANGÉLICA LISSETH BLANCO CONCHA','ING_SIST 128CSIS-1');
--Fila 9
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070021,'ANGÉLICA MARIA ROCHA GARCIA','ING_SIST 128CSIS-1');
--Fila 10
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070022,'ANGIE TATIANA FERNÁNDEZ MARTÍNEZ','ING_SIST 128CSIS-1');
--Fila 11
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070023,'TATIANA ANGIE MARTÍNEZ FERNÁNDEZ ','ING_SIST 128CSIS-1');
--Fila 12
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070024,'CAMILO VILLAMIZAR ARISTIZABAL','ING_IND 128CSIS-2');
--Fila 13
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070025,'CAMILO RODRÓGUEZ BOTERO ','ING_IND 128CSIS-2');
--Fila 14
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070026,'CAMILO ALBERTO CORTÉS MONTEJO','ING_IND 128CSIS-2');
--Fila 15
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070027,'CAMILO ENRIQUE GOMEZ RODRIGUEZ','ING_IND 128CSIS-2');
--Fila 16
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070028,'CARLOS ANDRÉS POLO CASTELLANOS ','ING_IND 128CSIS-2');
--Fila 17
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070029,'CARLOS DIDIER CASTAÑO CONTRERAS','ING_CVL 128CSIS-3');
--Fila 18
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070030,'CAROL RUCHINA GOMEZ GIANINE','ING_CVL 128CSIS-3');
--Fila 19
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070031,'CAROLINA PINTOR PINZON','ING_CVL 128CSIS-3');
--Fila 20
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070032,'CATHERINE OSPINA ALFONSO','ING_CVL 128CSIS-3');
--Fila 21
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070033,'CINTHYA FERNANDA DUSSEP GUZMÁN ','ING_CVL 128CSIS-3');
--Fila 22
INSERT INTO ALUMNOS (NO_MATRICULA, NOMBRE, GRUPO_CLASE) VALUES (100070034,'CLAUDIA LILIANA TORRES FRIAS ','ING_CVL 128CSIS-3');
--Fila 23

```

Figura 7.

Hoja de Trabajo

Generador de Consultas

SELECT * FROM ALUMNOS;

Salida de Script x

Resultado de la Consulta x

SQL

Todas las Filas Recuperadas: 23 en 0,004 segundos

	NO_MATRICULA	NOMBRE	GRUPO_CLASE
1	100070012	ADRIANA CAROLINA HERNANDEZ	ING_SIST_128CSIS-1
2	100070013	ADRIANA MARCELA REY SANCHEZ	ING_IND 128CSIS-2
3	100070014	ALEJANDRO ABONDANO ACEVEDO	ING_CVL 128CSIS-3
4	100070015	ALEJANDRO ABONDANO ACEVEDO	ING_SIST 128CSIS-1
5	100070016	ANDREA CATALINA ACERO CARO	ING_SIST 128CSIS-1
6	100070017	ANDREA LILIANA CRUZ GARCIA	ING_SIST 128CSIS-1
7	100070018	ANDRES FELIPE VILLA MONROY	ING_SIST 128CSIS-1
8	100070020	ANGELICA LISSETH BLANCO	ING_SIST 128CSIS-1
9	100070021	ANGELICA MARIA ROCHA	ING_SIST 128CSIS-1
10	100070022	ANGIE TATIANA FERNANDEZ M	ING_SIST 128CSIS-1
11	100070023	TATIANA ANGIE MARTÓNEZ	ING_SIST 128CSIS-1
12	100070024	CAMILO VILLAMIZAR ARISTIZ	ING_IND 128CSIS-2
13	100070025	CAMILO RODRÓGUEZ BOTERO	ING_IND 128CSIS-2
14	100070026	CAMILO ALBERTO CORT	ING_IND 128CSIS-2
15	100070027	CAMILO ENRIQUE GOMEZ	ING_IND 128CSIS-2
16	100070028	CARLOS ANDRÉS POLO C	ING_IND 128CSIS-2
17	100070029	CARLOS DIDIER CAST	ING_CVL 128CSIS-3
18	100070030	CAROL RUCHINA GOMEZ	ING_CVL 128CSIS-3
19	100070031	CAROLINA PINTOR P	ING_CVL 128CSIS-3
20	100070032	CATHERINE OSPINA	ING_CVL 128CSIS-3
21	100070033	CINTHYA FERNANDA	ING_CVL 128CSIS-3
22	100070034	CLAUDIA LILIANA TORRES	ING_CVL 128CSIS-3

Figura 8.

Sentencias para consultas en ORACLE

Seleccionamos todo lo que hay en la tabla *alumnos* donde el valor de la columna *nombre* sea todos *nombre* que comienzan con la inicial A.

```
select * from alumnos where nombre like "a%";
```

Seleccionamos todos los datos de la comuna *alumnos* pero un poco más ordenas de la consulta mostrada anteriormente

```
SELECT no_matricula,nombre,grupo_clase,fecha,calificacion,controles_escritos_no_control FROM alumnos INNER JOIN  
alumnos_controles_escritos ON alumnos.no_matricula = alumnos_controles_escritos.alumnos_no_matricula
```

Seleccionamos *nombre* y *apellido* de la tabla *profesores* y esto nos regresa el nombre y apellido de todos los profesores registrados en la tabla *profesores*.

```
SELECT nombre,apellido FROM profesores;
```

Seleccionamos el nombre y apellido, utilizando AS le podemos cambiar por cualquier otro parámetro donde no afectará esos cambios en la base de datos solo se mostrará el cambio cuando se ejecuta la consulta

```
SELECT nombre AS name ,apellido AS surname FROM profesores;
```

Seleccionamos todo de la tabla *acceso* donde el valor de la columna *nombre* sea todos los nombres que inician con la letra *a* y el rol sea *alumno*.

```
SELECT * FROM acceso WHERE nombre LIKE "a%" AND rol = "alumno"
```

Seleccionamos todo de la tabla *acceso* donde el valor de la columna *nombre* sea todos los nombres que inician con la letra *Y* y el rol sea *profesor*.

```
SELECT * FROM acceso WHERE nombre LIKE "y%" AND rol = "profesor"
```

Repositorio GitHub

- https://github.com/jmontiel02/Apb_examenes

Vídeos Jesús Montiel y Boris Pérez respectivamente.

- https://drive.google.com/file/d/1S9R2RtGTX55-dRn_LLpCpYUzl0ODua1hh/view?usp=sharing
- <https://youtu.be/xZr0D4YuXik>