

EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: COMPONENTES DE UNA BASE DE DATOS RELACIONAL.
- EXERCISE 2: CARDINALIDAD DE LAS RELACIONES.
- EXERCISE 3: INSTALACIÓN Y CONFIGURACIÓN DEL MOTOR DE BASE DE DATOS.
- EXERCISE 4: SQL SHELL (PSQL).

EXERCISE 1: COMPONENTES DE UNA BASE DE DATOS RELACIONAL.

Dentro de una base de datos relacional, encontramos los siguientes componentes:

- **Base de Datos (BD):** representa toda la estructura física de una base de datos, es decir, tanto la implementación del modelo E-R, como el universo de los datos que contiene.
- **Tabla:** representa la unidad fundamental de una base de datos relacional, la cual contiene una representación física de todos los atributos; a un elemento de la tabla se le llama registro, que es una fila dentro de una tabla.

Una tabla debe tener definido como base lo siguiente:


- **Nombre:** nombre único de la tabla para la base de datos que se está implementado desde el modelo E-R.
- **Atributos:** se deben implementar todos los atributos diseñados en el modelo E-R, indicar el tipo de datos, si forma parte de la PK, y si puede contener valores nulos.

El siguiente cuadro representa el esquema de la tabla Clientes, el cual a su vez, es la representación de la entidad Clientes definida en el modelo E-R.

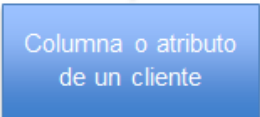
Clientes		
	Atributo	Tipo de Dato
PK	Rut	Alfanumérico
	Nombre	Alfanumérico
	Correo	Alfanumérico
	Giro	Alfanumérico
	Dirección	Alfanumérico

El siguiente cuadro representa los datos de la tabla Clientes (Matriz que representa los datos físicos contenidos en la tabla clientes de una BD)

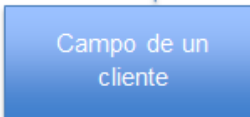
Clientes				
Rut	Nombre	Correo	Giro	Dirección
11.111.111-1	Carlos	carlos@correo.cl	Particular	Los Nogales
22.222.222-2	Eduardo	eduardo@correo.cl	Comerciante	Eleuterio Ramirez
33.333.333-3	Pedro	pedro@correo.cl	Ferretería	San Diego



Fila o Registro de un cliente



Columna o atributo de un cliente



Campo de un cliente

A partir de este cuadro, podemos entender lo que es una Fila, Columna y Campo, desde el punto de vista de los datos.

- **Atributo:** elemento de una tabla que describe un dato de cada uno de los registros ésta; debe contar con las siguientes definiciones básicas:
 - *Nombre:* cada atributo debe tener un nombre único dentro de cada tabla.
 - *Tipos de datos:* indica qué tipo de información se almacenará en cada atributo; por ejemplo: Alfanumérico, Numérico, Fecha, entre otros. Generalmente, para cada tipo de dato se deben indicar sus dimensiones, en el caso de Alfanumérico(100), indica que se podrán almacenar como máximo 100 caracteres.
 - *Permitir nulos:* indica si el dato puede o no almacenar un valor nulo "null".

- **PK:** se debe indicar si el atributo es o forma parte de la llave primaria.
- **Relaciones:** las relaciones del modelo E-R, también se pueden implementar físicamente en una BD, se hace en los siguientes casos:
 - Cuando queremos que la BD controle la integridad de los datos, sobre la base del modelo definido; si los datos que se insertan o modifican en la BD no respetan las restricciones de una relación, se produce un error generado por el motor de datos.
 - Cuando queremos generar consultas optimizadas, ya que el motor de datos, al conocer de antemano las relaciones existentes entre las diferentes tablas, puede elaborar planes de ejecución óptimos.
- **Índices:** son uno o más atributos, por los cuales se indexan los pertenecientes al índice; la indexación se produce al momento de agregar o modificar un registro, su ventaja es que la búsqueda y ordenación de registros por los atributos indexados es muy eficiente, ya que el motor de datos tiene el trabajo hecho de antemano. Éstos pueden o no, contener registros duplicados según su tipo. Una PK es un índice que no permite duplicados, y por lo mismo se denomina **llave primaria**. Dentro de una misma tabla, se pueden definir otros índices sin duplicados, pero solo uno será la llave primaria, y además, definir varios índices con duplicados.

EXERCISE 2: CARDINALIDAD DE LAS RELACIONES.

La cardinalidad es el número de entidades con la cual otra entidad se puede asociar mediante una relación.

Ésta puede ser:

- **Uno a uno:** (1:1) dada una entidad A relacionada con una entidad B; para cada elemento de A existe un elemento de B, y para cada elemento de B existe un elemento de A.
- **Uno a muchos:** (1:N) dada una entidad A relacionada con una entidad B; para cada elemento de A existen N elementos de B, y para cada elemento de B existe un elemento de A.
- **Muchos a uno:** (N:1) dada una entidad A relacionada con una entidad B; para cada elemento de A existe un elemento de B, y para cada elemento de B existen N elementos de A.

- **Muchos a muchos:** (N:M) dada una entidad A relacionada con una entidad B; para cada elemento de A existen N elementos de B, y para cada elemento de B existen M elementos de A.

El tipo de cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: "1:1", "1:N" y "N:M", aunque la notación depende del lenguaje utilizado, la que más se usa actualmente es el unificado.

Otra forma de expresar la cardinalidad, es situando un símbolo cerca de la línea que conecta una entidad con una relación:

- 0: si cada instancia de la entidad no está obligada a participar en la relación.
- 1: si toda instancia de la entidad está obligada a participar en la relación y, además, solamente lo hace una vez.
- "N", "M", o "*": si cada instancia de la entidad no está obligada a participar en la relación, y puede hacerlo cualquier número de veces.



- Un cliente puede efectuar 0 o N compras (venta), y una compra (venta) la puede hacer solo un cliente (relación directa).
- Una venta puede tener 1 o muchos Productos, y un Producto puede estar en 0 o muchas ventas (relación directa).
- Un cliente puede comprar (venta) 0 o muchos productos, y un producto puede ser vendido a muchos clientes (relación indirecta).

EXERCISE 3: INSTALACIÓN Y CONFIGURACIÓN DEL MOTOR DE BASE DE DATOS.

Dentro del mercado podemos encontrar diferentes motores de bases de datos relacionales, entre los que podemos destacar se encuentran:

- PostgreSQL.
- MySQL.
- Oracle.
- SQL Server.

Estos sistemas de base de datos relacional admiten todas las características principales de SQL. PostgreSQL y MySQL son admitidos por los principales sistemas operativos, como: Windows, Linux y MacOS; mientras que Oracle soporta muchos otros idiomas, y variados sistemas operativos lo admiten. Todos presentan la característica de poseer Licencia Abierta (Open Source), menos SQL Server que, además, solo es admitido por Windows.

En Oracle, en la ejecución de las consultas o comandos, los cambios se realizan solo en la memoria. No se confirma ningún cambio hasta que el DBA (Administrador de base de datos) emita un comando **COMMIT** explícito. Tan pronto como se ejecuta, los cambios se realizan en el disco y en el comando después de que **COMMIT** comience una nueva transacción. PostgreSQL y MySQL poseen la opción de **AUTOCOMMIT**, que viene activada, y los cambios, por defecto, se realizan en el disco.

A la hora de elegir entre un sistema y otro, muchos usuarios se hacen una pregunta: ¿cuál es más fácil de usar? En esta situación la respuesta no es muy precisa, ya que tanto MySQL, como PostgreSQL, se consideran sistemas de bases de datos sencillos.

PostgreSQL brinda opciones más complejas que MySQL, ya que suele estar orientado para bases de datos más grandes, y con consultas más largas. Algunas de sus funciones, como la de unir tablas, hacen que sea mejor valorado por algunos desarrolladores frente a su clásico rival.

Si nos dirigimos al sitio oficial de [PostgreSQL](#), podremos observar las novedades que trae, y las diferentes versiones disponibles para descargar.

PostgreSQL: The World's Most Advanced Open Source Relational Database

[Download →](#)[New to PostgreSQL?](#)

New to PostgreSQL?

PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

There is a wealth of information to be found describing how to *install* and use PostgreSQL through the official documentation. The PostgreSQL community provides many helpful places to become familiar with the technology, discover how it works, and find career opportunities. Reach out to the community [here](#).

[Learn More](#)[Feature Matrix](#)

Latest Releases

2021-08-12 - PostgreSQL 13.4, 12.8, 11.13, 10.18, 9.6.23, and 14 Beta 3 Released!

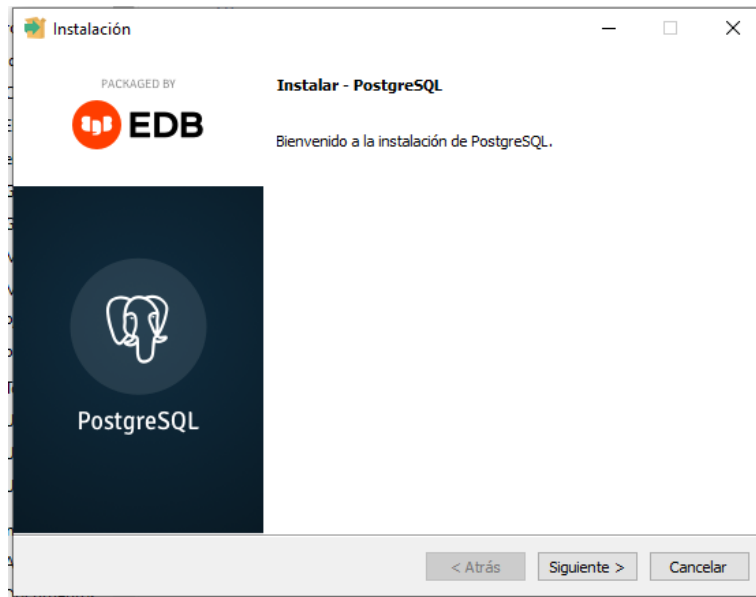
The PostgreSQL Global Development Group has released an update to all supported versions of our database system, including 13.4, 12.8, 11.13, 10.18, and 9.6.23, as well as the third beta release of PostgreSQL 14. This release closes one security vulnerability and fixes over 75 bugs reported over the last three months.

PostgreSQL 9.6 will stop receiving fixes on November 11, 2021. If you are running PostgreSQL 9.6 in a production environment, we suggest that you make plans to upgrade to a newer, supported version of PostgreSQL.

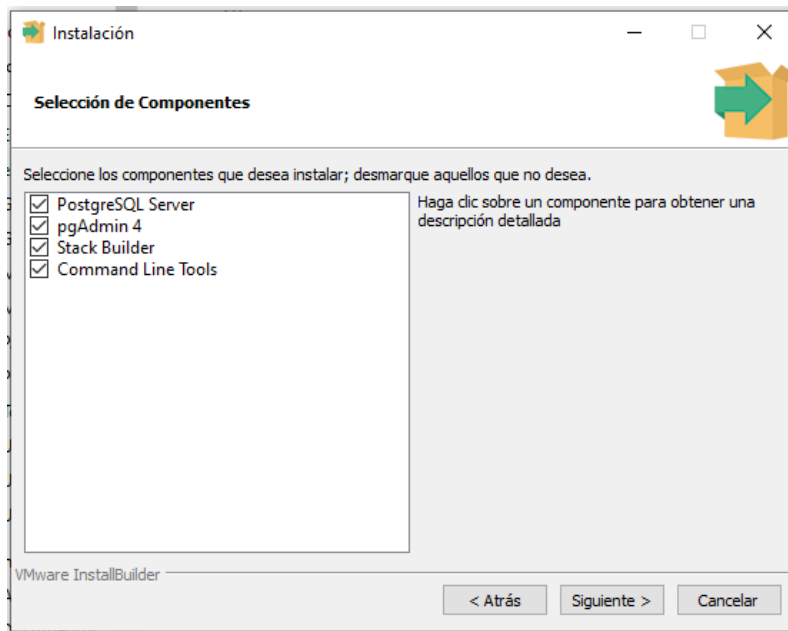
In the spirit of the open source PostgreSQL community, we strongly encourage you to test the new features of PostgreSQL 14 in your systems to help us eliminate bugs or other issues that may exist. While we do not advise you to run PostgreSQL 14 Beta 3 in your production environments, we encourage you to find ways to run your typical application workloads against this beta release.

[13.4 · 2021-08-12 · Notes](#)[12.8 · 2021-08-12 · Notes](#)[11.13 · 2021-08-12 · Notes](#)[10.18 · 2021-08-12 · Notes](#)[9.6.23 · 2021-08-12 · Notes](#)[Download](#)[Why Upgrade?](#)[Security](#)

Después de descargarlo, debes abrir el archivo, e instalarlo con permisos de administrador:



Presionamos “Siguiete”, y elegimos la carpeta de destino de la instalación.



Seleccionamos los componentes a instalar, y presionamos “Siguiete” hasta terminar todo el proceso.



Por último, presionamos “Terminar”, y ya tenemos instalado nuestro motor de bases de datos relacional y sus componentes más importantes: PgAdmin4 y SQL Shell (psql).

EXERCISE 4: SQL SHELL (PSQL).

Psql es una interfaz basada en terminales de entrada/salida para PostgreSQL. Le permite escribir consultas de forma interactiva, enviarlas a PostgreSQL, y ver sus resultados. Alternativamente, la entrada puede ser de un archivo, o de argumentos de línea de comando. Además, psql proporciona una serie de metacomandos y varias funciones de tipo Shell para facilitar la escritura de scripts, y automatizar una amplia variedad de tareas.

Para ingresar a SQL Shell, basta con colocar “psql” en el menú de Windows, y seleccionar la primera opción; una vez dentro, ingresaremos a la base de datos presionando la tecla enter en Server, Database, Port y Username, luego, ingresaremos nuestra contraseña y tendremos la siguiente vista.


```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:
psql (13.4)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.

postgres=#
```

1. Alistar las bases de datos disponibles.

Puede utilizar el comando `\l` para obtener una lista de todas las bases de datos disponibles.

```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Contraseña para usuario postgres:
psql (13.4)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.

postgres=# \l
          Listado de base de datos
+-----+-----+-----+-----+-----+-----+
| Nombre | Dueño | Codificaci| Collate | Ctype | Privilegios |
+-----+-----+-----+-----+-----+-----+
| postgres | postgres | UTF8 | Spanish_Chile.1252 | Spanish_Chile.1252 |  |
| primera | postgres | UTF8 | Spanish_Chile.1252 | Spanish_Chile.1252 |  |
| template0 | postgres | UTF8 | Spanish_Chile.1252 | Spanish_Chile.1252 | =c/postgres +
|          |          |          |          |          | postgres=CTc/postgres |
| template1 | postgres | UTF8 | Spanish_Chile.1252 | Spanish_Chile.1252 | =c/postgres +
|          |          |          |          |          | postgres=CTc/postgres |
(4 filas)

postgres=#
```

Como se puede ver, se cuenta con las siguientes bases de datos:

- postgres
- primera

Se puede ignorar el resto de las entradas. Ahora, veamos la lista de tablas disponibles en la base de datos actual.

2. Alistar las tablas disponibles en la base de datos actual.

El comando `\dt` se encarga de esto.

```
postgres=# \dt
No se encontró ninguna relación.
postgres=#
```

Pero desafortunadamente, la base de datos de Postgres no tiene ninguna tabla creada en ella (las tablas se conocen como relaciones en la literatura de administración de bases de datos). Cambiemos a otra base de datos, y veamos si tiene alguna tabla creada en ella.

3. Cambiar a otra base de datos.

La sintaxis para hacer esto es: `\c database_name`. Supongamos que desea cambiar a una base de datos llamada `primera`, puede hacerlo de la siguiente manera:

`\c primera`

```
postgres=# \dt
No se encontró ninguna relación.
postgres=# \c primera
Ahora está conectado a la base de datos «primera» con el usuario «postgres».
primera=#
```

Ahora, usando el comando `\dt`, veamos si tiene tablas:

```
primera=# \dt
          Listado de relaciones
 Esquema |          Nombre          | Tipo | Dueño
-----+-----+-----+-----
 public | directorio_telefonico | tabla | postgres
(1 fila)
```

Como puede ver, la base de datos “primera” tiene la siguiente tabla:

- directorio_telefonico

Supongamos que desea ver los detalles estructurales de una tabla en particular. Esto a menudo se denomina: descripción de una tabla. Veamos cómo se puede hacer.

4. Creando Base de Datos.

Para crearla, debemos escribir el comando: `CREATE DATABASE nombre_bd;`

```
primera=# create database segunda;  
CREATE DATABASE  
Duración: 224,096 ms
```

5. Creando Usuario/Rol.

Para ello, debemos escribir el comando: `CREATE USER nombre_user;`

```
primera=# CREATE USER ADMIN;  
CREATE ROLE  
Duración: 1,457 ms
```

6. Describiendo una tabla en particular.

La sintaxis general para hacerlo es: `\d nombre_tabla`. Supongamos que está en la base de datos “primera”, y desea describir la tabla denominada: “directorio_telefonico”. El comando para esto sería: `\d directorio_telefonico`.

```
primera=# \d directorio_telefonico
          Tabla %public.directorio_telefonico
+-----+-----+-----+-----+-----+
| Columna | Tipo          | Ordenamiento | Nulable | Por omisi|
+-----+-----+-----+-----+-----+
| nombre  | character     |               |         |          |
| apellido| character     |               |         |          |
| numero_telefonico | character     |               | not null |          |
| direccion | character     |               |         |          |
| edad    | integer       |               |         |          |
+-----+-----+-----+-----+-----+
Indices:
"directorio_telefonico_pkey" PRIMARY KEY, btree (numero_telefonico)
```

Como puede ver, "**\d**" brinda mucha información valiosa sobre la tabla, como los nombres de sus columnas, sus tipos de datos, modificadores de columnas, entre otros. Es posible que desee conocer la versión actual de su motor PostgreSQL. ¿Cómo hacerlo?

7. Conociendo la versión de PostgreSQL.

SELECT version(); te permite hacer esto.

```
primera=# select version();
          version
+-----+
 PostgreSQL 13.4, compiled by Visual C++ build 1914, 64-bit
(1 fila)
```

Ahora, supongamos que ha olvidado el último comando que ejecutó en el shell psql, y era importante. ¡No hay problema! Se puede recuperar fácilmente.

8. Ver el comando ejecutado anteriormente.

\g está ahí para hacer esto.

```
primera=# \g
          version
+-----+
 PostgreSQL 13.4, compiled by Visual C++ build 1914, 64-bit
(1 fila)
```

Como puede ver, `\g` ejecutó automáticamente el comando anterior. Ahora, si se desea revisar la lista de todos los comandos psql, disponibles para el motor PostgreSQL que está ejecutando; se verá a continuación.

9. Alistando todos los comandos disponibles.

Se puede obtener la lista de todos los comandos psql disponibles con `\? -`.

```
SQL Shell (psql)

primera=# \?
General
  \copyright      mostrar términos de uso y distribución de PostgreSQL
  \crosstabview [COLUMNAS] ejecutar la consulta y desplegar en %crosstab
  \errverbose      mostrar error más reciente en máxima verbosidad
  \g [(OPTIONS)] [FILE] ejecuta la consulta (y envía el resultado a un fichero o |pipe);
                   \g sin argumentos es equivalente a un punto y coma
  \gdesc          describir resultado de la consulta, sin ejecutarla
  \gexec          ejecutar la consulta, luego ejecuta cada valor del resultado
  \gset [PREFIXO] ejecutar la consulta y almacenar los resultados en variables
                   de psql
  \gx [(OPTIONS)] [FILE] como \g, pero fuerza el modo de salida expandido
  \q              salir de psql
  \watch [SEGS]   ejecutar consulta cada SEGS segundos

Ayuda
  \? [commands]   desplegar ayuda sobre las %rdenes backslash
  \? options       desplegar ayuda sobre opciones de línea de %rdenes
  \? variables     desplegar ayuda sobre variables especiales
  \h [NOMBRE]      mostrar ayuda de sintaxis de %rdenes SQL;
                   use %* para todas las %rdenes

B·fer de consulta
  \e [ARCHIVO] [L=NEA]
                   editar el b·fer de consulta (o archivo) con editor externo
  \ef [NOMBRE-FUNCIÓN [L=NEA]]
                   editar una función con editor externo
  \ev [NOMBRE-VISTA [L=NEA]]
                   editar definición de una vista con editor externo
```