{desafío}
latam_

Base de datos relacionales _

Parte I

Introducción a las bases de datos

Competencias

- Reconocer el rol de las bases de datos relacionales.
- Reconocer las características de una RDBMS.
- Aprender qué es el lenguaje estructurado de consultas (SQL).

Bases de datos y RDBMS

- Las bases de datos se definen como un conjunto de información relacionada que se encuentra ordenada o estructurada.
- Bases de datos relacionales y no relacionales.
- RDBMS: Sistema de Gestión de Bases de datos Relacionales

Algunas de sus características son:

- El lenguaje de datos para realizar consultas a la base de datos es SQL
- Almacena datos en tablas
- Persiste los datos en forma de filas y columnas
- Recupera datos de una o más tablas
- Crea índices para una recuperación de datos más rápida

¿Qué es SQL?

- Structured Query Language (Lenguaje estructurado de consultas) es un lenguaje creado para la definición y la manipulación de bases de datos relacionales.
- El beneficio de este lenguaje es facilitar la administración de datos almacenados.
- El lenguaje SQL está compuesto por cláusulas, operadores y funciones de agregado.

Tipos de bases de datos

Según la variabilidad de los datos	Se refiere a cómo se estructuran los datos y su grado de modificación en el tiempo. Existen dos tipos: estáticas y dinámicas.
Según el contenido	Se refiere al tipo de datos que estamos almacenando. Por ejemplo: Bibliográficas, directorios, científicas, etc.
Según el modelo	Tiene relación a cómo se estructura la forma en la que guardan sus datos (descripciones), sus métodos de almacenamiento y recuperación.

Instalación y configuración de PostgreSQL

{desafío} latam_

Competencias

- Reconocer las ventajas de utilizar PostgreSQL.
- Instalar y configurar PostgreSQL en los distintos sistemas operativos.

¿Qué es PostgreSQL?

- “Un sistema de gestión de base de datos objeto-relacional, distribuido bajo licencia BSD y con código fuente disponible libremente. Es uno de los sistemas de gestión de base de datos más potente del mercado”. (*Autor: Zea, Molina y Redrován, 2017)*
- PostgreSQL es un sistema de base de datos relacional de alta disponibilidad, es decir, es estable, consistente y tolerante a fallos.

Ventajas y desventajas de PostgreSQL

Ventajas de PostgreSQL	Desventajas de PostgreSQL
Licencia gratuita	Alto consumo de recursos
Disponible para distintos sistemas operativos, como Windows, Linux y Unix, 32 y 64 bits.	Algunos comando o sentencias pueden ser poco intuitivas
Bajo mantenimiento	No tiene soporte en línea. Tiene foros oficiales y una gran comunidad que responde a las dudas.
Estabilidad, no se presentan caídas de bases de datos	
Alto rendimiento	
Gran capacidad de almacenamiento	
Gran escalabilidad, se ajusta al número de CPU y memoria disponible de forma óptima, soportando gran cantidad de peticiones simultáneas	

Instalación de PostgreSQL

Un aspecto a considerar es que la instalación de PostgreSQL dependerá del sistema operativo que se esté utilizando:

- Windows
- Linux
- MacOS

Administrar usuarios y bases de datos

Competencias

- Crear usuarios y asignar permisos para administrar una base de datos.
- Crear bases de datos para almacenar información de forma organizada.

Características de PostgreSQL

Sintaxis de PostgreSQL:

- Las instrucciones deben ser terminadas con el símbolo " ; " (punto y coma).
- Tener en cuenta que existen palabras reservadas para variables, ya que estos suelen ser comandos y pasa a ser fácil de confundir.
- No es sensible a las letras mayúsculas/minúsculas.

ALIAS	AND	AS
CREATE	CREATEDB	CREATEUSER
DATABASE	FROM	INNER
JOIN	LARGE	PASSWORD
WHERE	INSERT	UPDATE

Crear usuarios

- Entrar a la base de datos con el comando:

```
psql
```

- Crear un usuario:

```
CREATE USER nombre_usuario;
```

- Poner límites a un usuario:

```
CREATE USER nombre_usuario WITH comando_opcional;
```

Crear usuarios

Comandos posibles, para poner límites:

- PASSWORD
- ENCRYPTED PASSWORD
- UNENCRYPTED PASSWORD
- VALID UNTIL
- CREATEDB
- NOCREATEDB
- SUPERUSER
- NOSUPERUSER

```
CREATE USER Bastian WITH PASSWORD
'contraseña_secreta' VALID UNTIL
'2019-12-31';
```

Eliminar usuarios

- Eliminar usuarios sin tener que esperar la fecha de expiración:

```
DROP USER nombre_usuario;
```

- Consultas los usuarios que tienen acceso a la base de datos:

```
SELECT nombre_usuario FROM pg_user;
```

- Crear un superusuario o superuser

```
CREATE USER nombre_usuario WITH SUPERUSER;
```

Permisos para los usuarios

- Para dar acceso a todos los privilegios de una base de datos:

```
GRANT ALL PRIVILEGES ON DATABASE
database_name TO nombre_usuario;
```

- Para el caso de querer transformar al usuario en superuser:

```
ALTER USER myuser WITH SUPERUSER;
```

- Para dar permiso de creación de una base de datos:

```
ALTER USER nombre_usuario CREATEDB;
```

- Remover el superusuario:

```
ALTER USER username WITH NOSUPERUSER;
```

Crear una base de datos

- Para crear una base de datos, utilizar el siguiente comando:

```
CREATE DATABASE nombre_base_de_datos;
```

- Cambiar de base de datos:

```
\c nombre_base_de_datos
```

Eliminar una base de datos

- Para eliminar la base de datos utilizamos el comando:

```
DROP DATABASE nombre_base_de_datos;
```

Operaciones comunes a nivel de consola

Comando	Acción
\c nombre_base	Conectarse a una base de datos específica
\l	Listar todas las bases de datos existentes
\du	Listar todos los usuarios en el motor
\d	Listar todas las relaciones (o tablas) existentes en una base de datos específica
\dt	Lista todas las tablas de una base de datos
\q	Salir de la consola de PostgreSQL
\h	Mostrar la lista de comandos

Elementos de una base de datos

Competencias

- Interpretar el concepto de tabla en una base de datos.
- Reconocer cómo se relacionan las tablas mediante claves primarias y foráneas.
- Clasificar los tipos de datos que se pueden utilizar en una base de datos.

Bases de datos relacionales versus bases de datos no relacionales

Relacional	No Relacional
Cuando el volumen de datos no crece o lo hace gradualmente.	Cuando el volumen de datos crece muy rápidamente.
Cuando las necesidades de proceso se pueden asumir en un solo servidor o en N servidores definidos previamente.	Cuando las necesidades del proceso no se pueden prever.
Cuando no existen peaks de uso o son esporádicos.	Cuando existen peaks de uso en múltiples ocasiones.
Cuando requerimos mantener la integridad referencial.	Cuando la información no requiere mantener relación entre los registros.
Estructura de datos mayormente estática.	Estructura de datos variable.

Ejercicio guiado: Directorio telefónico

Crear un registro telefónico donde alojaremos el nombre, apellido, número telefónico, dirección y edad de una serie de individuos. Resulta que el registro en sí será la tabla **directorio_telefonico**. Además se desea incorporar la información de otra tabla llamada **agenda** que tiene las columnas nick y numero_telefonico

Para empezar a crear nuestra base de datos utilizaremos tablas

Tablas

Una base de datos se compone de múltiples tablas. Cada una de éstas presentarán dos dimensiones:

- Filas, que representan a los registros en la tabla.
- Columnas, que van a representar los atributos ingresados en cada registro, definiendo el tipo de dato a ingresar.

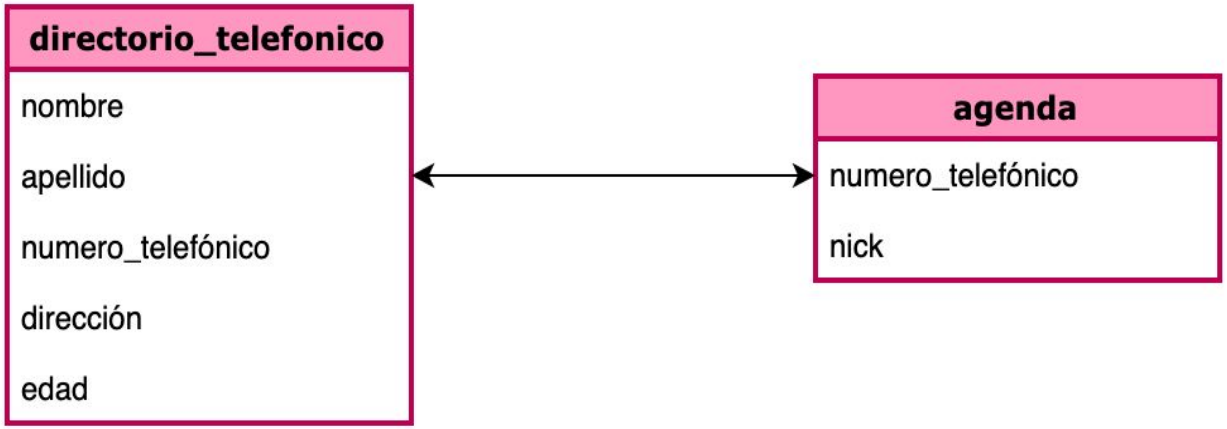
Clave primaria (primary key)

Cuando se hace referencia a una columna dentro de su tabla de origen, hablaremos de una clave primaria. Esta clave siempre será de carácter único.

Clave foránea (foreign key)

Cuando se hace referencia a una columna identificadora en otra tabla a la cual hacemos referencia, hablamos de una clave foránea.

Veamos un ejemplo



Tipos de datos

Instrucciones de creación, inserción, actualización y eliminación de datos

Competencias

- Construir bases de datos para el almacenamiento persistente de información.
- Crear tablas con sus atributos y tipos de datos correspondientes para el direccionamiento de datos.
- Crear claves primarias y foráneas para el enlace de referencias entre tablas.
- Importar fichero con extensión .sql

Creación de tablas

Para crear una tabla se debe utilizar el comando CREATE TABLE acompañado del nombre de la tabla y los atributos con su tipo de dato:

```
CREATE TABLE nombre_tabla(
    columna1 tipo_de_dato1,
    columna2 tipo_de_dato2,
    columna3 tipo_de_dato3,
    PRIMARY KEY (columnaN)
);
```

Ejemplo: Tabla **Directorio_telefonico**

```
CREATE TABLE Directorio_telefonico(nombre
VARCHAR(25), apellido VARCHAR(25),
numero_telefonico VARCHAR(8), direccion
VARCHAR(255), edad INT, PRIMARY KEY
(numero_telefonico) );
```

Creando una tabla con clave foráneas

Posterior a la creación de la primera tabla, definir los componentes de la segunda tabla **agenda** con los campo numero_telefonico y nick, asignando una clave foránea proveniente de la tabla directorio_telefonico.

```
    -- Creamos una tabla con el nombre agenda
    CREATE TABLE Agenda(
    -- Definimos el campo nick con el tipo de dato cadena con un largo de 25 caracteres
    nick VARCHAR(25),
    -- Definimos el campo numero_telefonico con el tipo de dato cadena con un largo de 8 caracteres.
    numero_telefonico VARCHAR(8),
    -- Vinculamos una clave foránea entre nuestra columna numero_telefonico y su símil en la tabla directorio telefónico
    FOREIGN KEY (numero_telefonico) REFERENCES Directorio_telefonico(numero_telefonico)
    );
```

Inserción de datos en una tabla

- Para que una base de datos sea útil, debe contener datos, para insertar datos a una tabla existe el comando INSERT.
- Esto es un proceso ordenado y debemos especificar a qué fila pertenecen los datos que estamos ingresando.
- La forma canónica de la instrucción INSERT es la siguiente:

```
INSERT INTO nombre_tabla (columna1, columna2, columna3) VALUES
(valor1, valor2, valor3);
```

Ingresar registros a tabla Directorio_telefonico

nombre	apellido	numero_telefonico	direccion	edad
Juan	Perez	12345678	Villa Pajaritos	21
Fabian	Salas	32846352	Playa Ancha	21
John	Rodriguez	23764362	Constitucion	21
Braulio	Fuentes	23781363	Rancagua	19
Pedro	Arriagada	38472940	Valdivia	23
Matias	Valenzuela	38473623	Nogales	22
Cristobal	Missana	43423244	Con Con	20
Farid	Zalaquett	32876523	La Florida	20
Daniel	Hebel	43683283	San Bernardo	20
Javiera	Arce	94367238	Quilpue	20

Ingresar registros a tabla Agenda

nick	numero_telefonico
Juanito	12345678
Juancho	12345678
Peter	38472940
Mati	38473623
Cris	43423244
Javi	94367238
Farid	32876523
Dani	43683283

Ejercicio propuesto

La empresa Ovalle Electronics SPA necesita una base de datos para almacenar sus productos y el historial de ventas del día a día. Crea una base de datos llamada OvalleElectronicsSPA con las siguientes tablas:

nombre	ID	fecha_creacion	proveedor	categoría
TV RV-25	2468	2020-08-16	Dende SPA	televisores

Tabla Productos

fecha	ID_Producto	cliente	metodo_pago	referencia
2021-02-01	2468	Bruce Lee	efectivo	34414
2020-11-15	2468	Chuck Norris	débito	43224

Tabla Ventas

Actualización de registros

- Para actualizar los datos de un registro, se debe utilizar el comando UPDATE de la siguiente forma:

```
UPDATE nombre_tabla SET columna1=valor_nuevo WHERE condicion;
```

- Para el ejemplo: Juan se cambió de casa a Villa Los Leones, por lo que se debe actualizar la tabla **Directorio_telefonico**:

Ejercicio propuesto

La empresa Ovalle Electronics SPA registró en la venta con referencia 43224 un método de pago como “débito” y en una auditoría realizada recientemente se comprobó que esa venta fue cancelada por el cliente con una tarjeta de “crédito”, por lo que se le pide al programador que se encarga de la base de datos hacer esa corrección. Usa el ejercicio propuesto 1 para este ejercicio.

Eliminación de registros

- La sintaxis para eliminar toda la tabla es:

```
DELETE FROM tabla;
```

- Para poder seleccionar qué registros queremos borrar debemos hacerlo de la siguiente forma:

```
DELETE FROM tabla WHERE condicion;
```

- Si quisiéramos borrar todos los datos de una tabla:

```
DELETE FROM Agenda;
```

- Del ejemplo eliminar a John de la tabla de Directorio_telefonico:

```
DELETE FROM Directorio_telefonico
WHERE nombre='John';
```

Ejercicio propuesto

La empresa Ovalle Electronics SPA decidió dejar de vender el televisor modelo RV-25, por lo que le pide al programador encargado de la base de datos que lo elimine del registro de los productos.

Lo pernicioso del método DELETE

- Sólo el administrador de la base de datos debería ser capaz de eliminar datos de una base de datos.
- Este comando es susceptible a errores, ya que si no implementamos correctamente la condición en el comando WHERE, podemos eliminar datos que no teníamos pensado borrar y no hay posibilidad de recuperarlos.

Añadir columnas

Si deseamos añadir una columna específica de una tabla, podemos implementar la siguiente sintáxis:

```
ALTER TABLE nombre_tabla
ADD nueva_columna tipo_de_dato;
```

Eliminar columnas

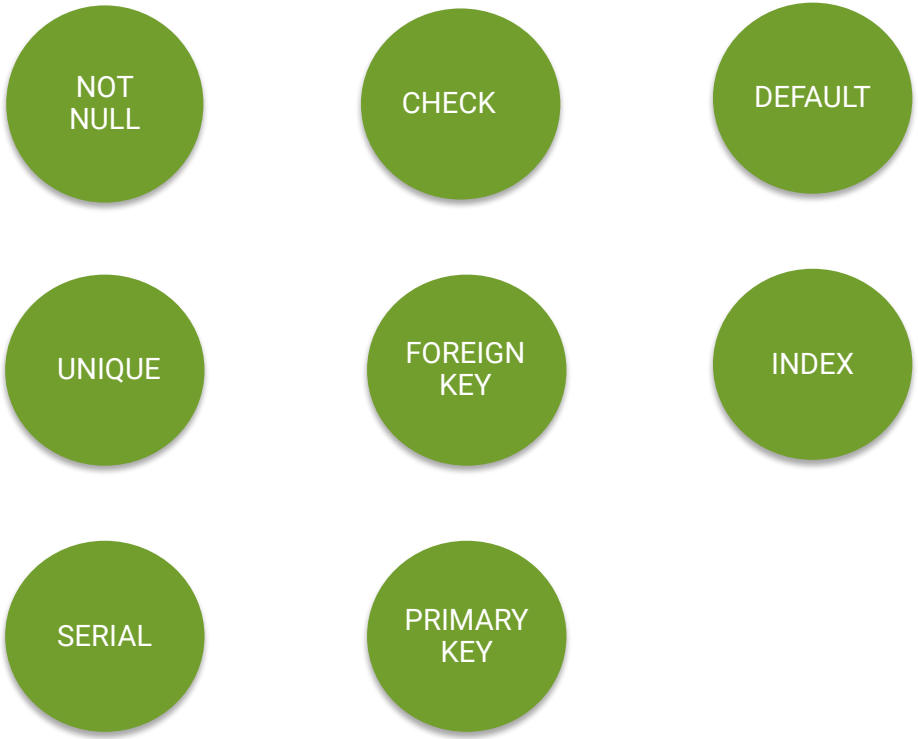
Si deseamos eliminar alguna columna en específico, podemos implementar la instrucción DROP de manera análoga a como lo hicimos con ADD.

```
ALTER TABLE nombre_tabla
DROP nueva_columna;
```

Ejercicio propuesto

La empresa Ovalle Electronics SPA decidió desestimar el almacenamiento de la fecha de creación de los productos por lo que pide que esa propiedad sea eliminada y además solicita la creación de una nueva propiedad llamada “color” para futuros filtros de búsqueda.

Restricciones



Restricciones

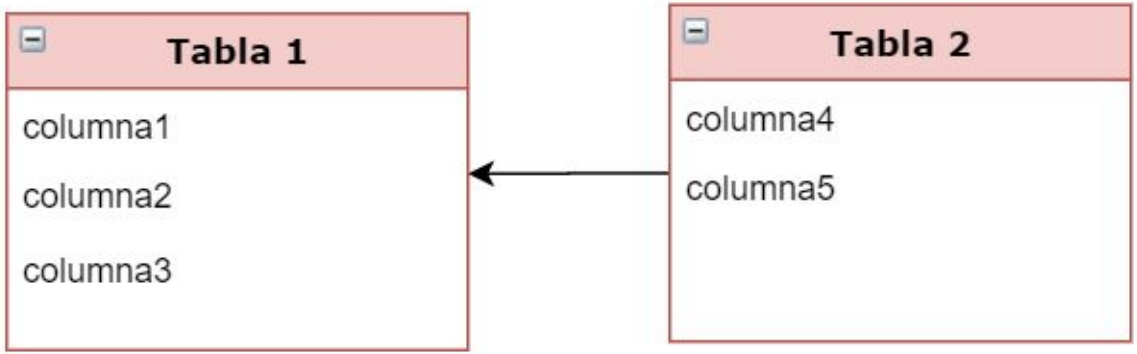
Las restricciones se aplican de la siguiente forma:

```
-- Creamos una tabla
CREATE TABLE nombre_tabla(
-- Declaramos una serie de restricciones a cada campo de dato creado
columna1 tipo_de_dato1 restriccion,
columna2 tipo_de_dato2 restriccion,
columna3 tipo_de_dato3 restriccion
);
```

Ejercicio propuesto

A la empresa Ovalle Electronics SPA le ha ido muy bien vendiendo televisores y ha empezado a contratar más personal por lo que consideró necesario tener en la base de datos un registro de sus empleados, almacenando su nombre, fecha de ingreso a la empresa, género y rut. Crea una tabla en la misma base de datos de los ejercicios propuestos, especificando que el rut es un valor único y que el nombre no puede ser un dato tipo null.

Restricciones a nivel de PRIMARY KEY y FOREIGN KEY



Restricciones a nivel de PRIMARY KEY y FOREIGN KEY

Suponiendo que la columna1 de la tabla 1 fuese su clave primaria y la columna4 de la tabla 2 la foránea, la forma de aplicarlas es la siguiente:

Nota que se ha agregado la restricción "UNIQUE" en la columna 1, para restringir que los valores de ese atributo en todos los registros deben ser únicos.

```
    CREATE TABLE tabla1(
    columna1 tipo_de_dato1 UNIQUE,
    columna2 tipo_de_dato2,
    columna3 tipo_de_dato3,
    PRIMARY KEY (columna1)
    );

    CREATE TABLE tabla2(
    columna4 tipo_de_dato4,
    columna5 tipo_de_dato5,
    FOREIGN KEY (columna4) REFERENCES
tabla1(columna1)
    );
```

Ejemplo Directorio telefónico

numero_telefonico será clave primaria (PRIMARY KEY) en Directorio_telefonico con la restricción “UNIQUE” para evitar repeticiones y al mismo tiempo existirá este atributo como la clave foránea (FOREIGN KEY) en Agenda.

```
CREATE TABLE Directorio_telefonico(
nombre VARCHAR(25),
apellido VARCHAR(25),
numero_telefonico VARCHAR(8) UNIQUE,
direccion VARCHAR(255),
edad INT,
PRIMARY KEY (numero_telefonico)
);

CREATE TABLE Agenda(
nick VARCHAR(25),
numero_telefonico VARCHAR(8),
nota VARCHAR(100),
FOREIGN KEY (numero_telefonico) REFERENCES
Directorio_telefonico(numero_telefonico)
);
```

Ejercicio propuesto

La empresa Ovalle Electronics SPA se dio cuenta que se están generando registros con datos que ya existen en otra tabla, y para evitar esta redundancia innecesaria le pide a su programador en base de datos que cree la relación entre las tablas Productos y Ventas. Vuelve a crear estas tablas asignando la clave primaria y foránea.

Eliminación de una tabla

- Si queremos eliminar la tabla Agenda, lo haremos de la siguiente manera:

```
DROP TABLE Agenda;
```

- La respuesta de PostgreSQL es:

```
DROP TABLE;
```

- Al consultar sobre la tabla Agenda, nos muestra lo siguiente:

```
select * from Agenda;

ERROR:  no existe la relación
«agenda»
LÍNEA 1: select * from agenda;
```

Ejercicio propuesto

La empresa Ovalle Electronics SPA ha bajado drásticamente sus ventas, no obstante ha notado en sus empleados un nivel técnico excelente y ha tomado la decisión de dejar de vender productos y cambiar su modelo de negocios para ofrecer servicio técnico, por lo que ya no necesita seguir usando el registro de productos, así que le pide a su programador que elimine la tabla de productos de la base de datos.

Truncado de una tabla

- Si lo que realmente necesitamos es eliminar sólo los registros, pero no la tabla en sí, se debe utilizar el comando TRUNCATE.
- Debes utilizar este comando sólo si es realmente necesario, ya que al igual que el comando DROP no es posible recuperar la información una vez eliminada.

```
TRUNCATE TABLE Agenda;
```

Ejercicio propuesto

Luego de un éxito tremendo ofreciendo servicio técnico la empresa Ovalle Electronics SPA ha logrado abrir una sucursal en otro estado del país, por lo que generó una copia de la base de datos para ser usada en esta nueva sede, pero ahora solicita al programador que vacíe la tabla de empleados para volver a llenarla pero con los nuevos trabajadores.

Cargar consultas desde un fichero

Crear ficheros con extensión .sql que permitirá escribir todos los comandos SQL para poder cargarlos.

La sintaxis para realizar esta operación es:

```
\i ubicación\nombre_fichero.sql
```

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com