Manipulación de datos y transaccionalidad en las operaciones

Sentencias para la manipulación de datos (Parte II)

{desafío} latam_

Utilizar lenguaje de manipulación de datos DML para la modificación de los datos existentes en una base de datos dando solución a un problema planteado

- Unidad 1:
  Bases de datos relacionales
- Unidad 2:
  Manipulación de datos y transaccionalidad en las operaciones
- Unidad 3:
  Definición de tablas
- Unidad 4:
  Modelos Entidad-Relación y Relacional

{desafío} latam_

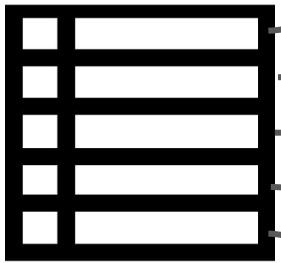
¿Qué aprenderás en esta sesión?

- *Utiliza sentencias de ingreso, actualización y borrado de registros utilizando lenguaje DML para manipular la información de tablas con integridad referencial de acuerdo a un modelo de datos existente.*

/* Tipos de datos */

Tipos de datos frecuentemente utilizados

Algunas ya utilizadas



- Tipos numéricos → INT/SMALLINT/BIGINT, FLOAT/DOUBLE
- Tipos de caracteres (caracteres, texto) → CHAR/VARCHAR
- Tipos temporales (fechas) → DATE/TIME/TIMESTAMP
- Tipos booleanos → BOOLEAN
- Varios más (arrays, json, uuid, etc)

Estos últimos no los cubriremos en este curso

Al momento de definir el tipo de dato evitamos que se guarden datos de tipo distintos, de esa forma "protegemos" los datos

/* Restricciones */

Las restricciones nos permiten agregar "protecciones" adicionales para cuidar los datos

Integridad de datos

Integridad de datos = correctitud de datos + completitud de los datos

- Al crear tablas, podemos añadir restricciones (en inglés constraints) a las columnas de una tabla para evitar que se ingresen datos que no cumplan ciertas condiciones.
- Las restricciones y el correcto uso de tipo de datos nos ayudan a evitar problemas de integridad de los datos
- En esta clase aprenderemos a evitar algunos de ellos.

Restricciones

Not Null, Unique

- **Not Null:** Al intentar insertar un valor nulo, ya sea directamente o porque no lo especificamos, obtendremos un error.

- **Unique:** Para asegurar que algún valor sea único, por ejemplo el correo de una persona o su RUT.

Restricciones

Primary Key y Foreign Key

- **Primary Key:** Es un tipo de restricción que impedirá que un dato pueda repetirse.

  Primary Key => Not Null + Unique

- **Foreign Key:** Estas claves son otro tipo de restricción que impedirán que se agregue un registro asociado a uno no existente, o que se borre un registro relacionado.

Ejercicios guiados

"Problemas de integridad" utiliza el “Material de apoyo - Restricciones not null y unique”

{desafío} latam_

Ejercicios guiados

Tenemos una tabla que contiene las siguientes columnas: nombre y apellido (utiliza el Material de apoyo - Restricciones not null y unique). Como podemos observar, la columna nombre tiene la restricción NOT NULL (la creación de columnas con restricción será abordado en la próxima unidad).

COLUMNS	DATA TYPE	CONSTRAINT
NOMBRE	TEXT	NOT NULL
APELLIDO	TEXT	
RUT	TEXT	UNIQUE

Ahora insertemos una fila de la siguiente manera:

```
INSERT INTO usuarios (apellido) VALUES ('Gómez');
```

En nuestra terminal obtendremos un error como el siguiente:

```
ERROR:  null value in column "nombre" violates not-null constraint
```

Ejercicios guiados

Ahora insertamos las siguientes filas

```
INSERT INTO usuarios (nombre, apellido, rut) VALUES ('Juan', 'Gonzalez', 'RUT1');

INSERT INTO usuarios (nombre, apellido, rut) VALUES ('Maria', 'Perez', 'RUT2',);

INSERT INTO usuarios (nombre, apellido, rut) VALUES ('Pedro', 'Fuentes', 'RUT2');
```

COLUMNS	DATA TYPE	CONSTRAINT
NOMBRE	TEXT	NOT NULL
APELLIDO	TEXT	
RUT	TEXT	UNIQUE

¿Qué problema existe en nuestros inserts?
Si observas bien, el segundo y tercer insert contienen el mismo rut, por lo tanto, no se cumple la restricción UNIQUE.

En la terminal observaremos un error como este:

ERROR: duplicate key value violates unique constraint "nombre_columna_key"
DETAIL: Key (nombre_columna)=(valor_duplicado) already exists.

Ejercicios guiados

Ahora, tenemos 2 tablas: una tabla **Autores** que contiene las columnas AutorID y NombreAutor y una tabla **Libros** que contiene las columnas LibroID, Titulo y AutorID (utiliza el Material de apoyo - Restricciones pk y fk).

En ambas tablas observamos que sus correspondientes ids corresponden al PK, es decir, a la clave primaria. Por otro lado, en la tabla Libros observamos que la columna AutorID es el FK o foreign key, la cual hace referencia a la tabla Autores

Tabla Autores

COLUMNS	DATA TYPE	CONSTRAINT
AutorID	INT	PK
NombreAutor	TEXT	

Tabla Libros

COLUMNS	DATA TYPE	CONSTRAINT
LibroID	INT	PK
Titulo	VARCHAR	
AutorID	INT	FK

Ejercicios guiados

- Insertemos un autor con un AutorID 2 el cual ya existe en la tabla Autores. Deberías recibir un error porque AutorID es una clave primaria y no puede repetirse.

  INSERT INTO Autores (AutorID, NombreAutor) VALUES (2, 'Nuevo autor');

- Insertemos un libro con un AutorID 1 que no existe en la tabla Autores. Deberías recibir un error debido a la restricción de clave foránea.

  INSERT INTO Libros (LibroID, Titulo, AutorID) VALUES (101, 'Libro Ejemplo', 1);

- Ahora, intentemos eliminar al autor con el AutorID 2, que tiene un libro asociado: deberías recibir un error que indica que la operación de eliminación viola la restricción de clave foránea

  DELETE FROM Autores WHERE AutorID = 2;

¿Por qué son importantes las restricciones?

/* Utilización de secuencias para asignar identificadores */

Identificadores auto incrementados

- Hasta el momento cuando definimos el campo ID en una tabla, dicho ingreso lo hacemos manualmente.
- En SQL es posible automatizar la definición de identificadores.
- Para automatizar podemos utilizar la sentencia SERIAL.

Veamos a través de un ejercicio cómo implementarlo.

Ejercicio guiado

"Autoincremento de IDs"

{desafío}
latam_

Autoincremento de IDs

- **Paso 1:** Creamos la base de datos restricciones_psql. Luego debemos conectarnos a ella.

```
create database restricciones_psql;
```

- **Paso 2:** Creamos una tabla llamada company con los campos id y nombre.

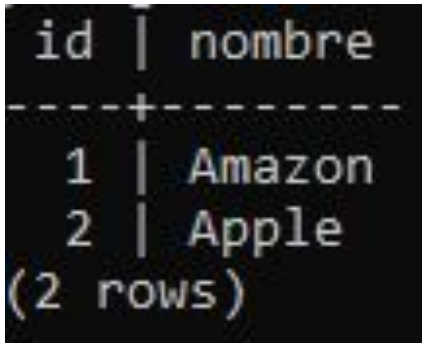
```
create table company (id serial primary key, nombre varchar not null unique);
```

Autoincremento de IDs

- **Paso 3:** Hacemos dos insert a la tabla.

```
insert into company(nombre) values('Amazon');
insert into company(nombre) values('Apple');
```

En el código a pesar de no definir el id este se ingresa automáticamente y de manera auto incremental.



Autoincremento de IDs

- **Paso 4:** Validar la restricción Unique dentro de la tabla. Para ello intentaremos ingresar un nombre que ya exista.

```
insert into company(nombre) values('Apple');
```

Al ejecutar veremos el siguiente error:

```
ERROR:  duplicate key value violates unique constraint "company_nombre_key"
DETAIL:  Key (nombre)=(Apple) already exists.
```

Este error comprueba que nuestra restricción(constraint) está funcionando.

Ejercicio propuesto

"Comprueba la restricción not null del campo nombre"

¿Por qué es importante la integridad referencial en las tablas?

{desafío} latam_

/* Insertar, actualizar y borrar datos con integridad referencial */

Ejercicio propuesto

"Agregando campos y operando sobre los registros"

{desafío} latam_

Contexto

A continuación, deberás incorporar los siguientes campos a la tabla de company, recuerda la instrucción `ALTER TABLE`:

- Dirección.
- Edad (años de servicio de la empresa).
- Nómina

Nota: Deberás asignar el tipo de dato correspondiente según el campo.

Para los años de servicio y la nómina utiliza Google para investigar este tipo de información.

{desafío} latam_

Contexto

Luego de agregar los campos:

- Elimina los datos de la tabla.
- Inserta nuevos registros donde se incluyan los nuevos campos agregados.
- Valida la restricción Not Null en todos los campos de la tabla.

¿Cuál fue el concepto que más te costó comprender?

{desafío} latam_

Próxima sesión...

- *Guía de ejercicios*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales