



Tema 2: Bases de datos relacionales



Objetivo

Transformar los modelos conceptuales a lógicos. Construir bases de datos para datos estructurados

- Utilización de los modelos en el mundo real 
- Transformación del modelado a la base de datos 



Temario

- Clases de entidades
- Modelo relacional, restricciones de integridad y normalización
- Arquitectura de los sistemas gestores de Bases de datos, OLTP, OLAP
- Lenguaje de consulta formal
- Lenguaje de consulta SQL



Logros

Capturar datos desde una pantalla de usuario real y almacenarlos en una base de datos. Manipulación de estos datos.



IMPORTANTE

- Para la construcción de los diagramas se hará uso de la herramienta que ofrece PgAdmin (ERD Project), hay otras opensource en evaluación, ya que PGModeler tiene problemas con el almacenamiento de los proyectos, lo cual afecta con la pérdida de los diagramas.
- El tema de herramientas de diagramas está en evaluación, la decisión de la herramienta a utilizar para la clase y proyectos de curso se informará el día 19 de febrero vía correo electrónico a lo que se adjuntará, un video y guías de instalación.
- Por el momento las actividades de diagramar en herramientas que sirvan para migrar datos van a quedar en pausa hasta que se determine la herramienta a utilizar si se indique el funcionamiento de esta.



Práctica SQL

SQL es el lenguaje estándar de definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo, ya que para este caso solo es necesario indicar que hacer más no como hacerlo.

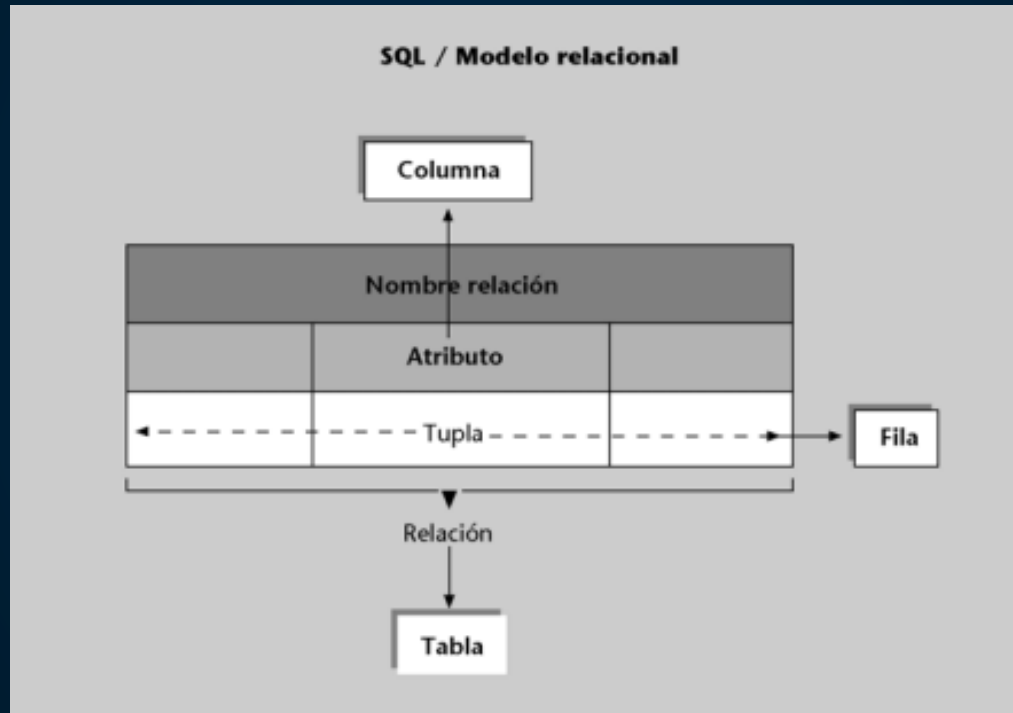
Este lenguaje ha presentado transformaciones a lo largo de la historia de su creación. Este lenguaje nació en los años setenta a manos de IBM recibiendo como primer nombre SEQUEL (**Structured English Query Language**), que más adelante fue reemplazado por SQL (**Structure Query Language**).

El estándar de manipulación de bases de datos ha pasado por varias versiones donde en un inicio permitía únicamente creación de tablas y llaves primarias y foráneas, luego se amplió para permitir agregar más tipos, nuevas operaciones, características de integridad y tablas temporales. Hoy en día es un lenguaje complejo de gran potencial.





Práctica SQL



Modelo con el cual trabaja el lenguaje SQL



Práctica SQL

Para ingresar sentencias de SQL, ingresa a la consola cmd de tu computador,

```
psql -U nombre_usuario;
```

```
C:\Users\erika\OneDrive\Desktop\Clases-Eafit\Bases-Datos-Practica>psql -U postgres
Password for user postgres:
psql (14.1)
WARNING: Console code page (437) differs from Windows code page (1252)
        8-bit characters might not work correctly. See psql reference
        page "Notes for Windows users" for details.
Type "help" for help.

postgres=#
```

Luego ingresa la contraseña asignada al momento de instalar postgres.

Algunos de los comandos de consola son propiamente de Postgres



Práctica SQL

Recuerda que:

Para listar todas las bases de datos disponibles debes usar el comando

```
\l
```

Para pasarse de una base de datos a otra, usar el commando

```
\c nombre_base de datos
```

La base de datos debe ser creada antes de ejecutar el commando \c



Práctica SQL

Si aún no tienes bases de datos creadas, la estructura o comando para crear una base de datos se define de la siguiente manera (Se toma ejemplo de clase 7)

```
CREATE DATABASE nombre_base_datos;
```




Práctica SQL

El comando o la sentencia de SQL para crear una tabla se estructura de la siguiente manera:

```
CREATE TABLE productos  
  (codigo_producto INTEGER,  
   nombre_producto CHAR(20),  
   tipo CHAR(20),  
   descripcion CHAR(50),  
   precio REAL,  
   PRIMARY KEY (codigo_producto));
```

Nombre de la tabla

Nombre de las columnas y tipo

Clave primaria



Práctica SQL

```
CREATE TABLE "USUARIO" (  
  cedula varchar(10) NOT NULL,  
  nombre varchar(20),  
  apellido varchar(20),  
  nombre_usuario varchar(10),  
  correo varchar(30),  
  clave varchar(8),  
  activo boolean,  
  CONSTRAINT pk_usuario PRIMARY KEY (cedula));
```

Antes de crear la tabla, revisa que elementos no deberían ser nulos



Práctica SQL

```
CREATE TABLE "USUARIO" (  
    cedula varchar(10) NOT NULL,  
    nombre varchar(20),  
    apellido varchar(20),  
    nombre_usuario varchar(10),  
    correo varchar(30),  
    clave varchar(8),  
    activo boolean,  
    CONSTRAINT pk_usuario PRIMARY KEY (cedula));
```

- Antes de crear la tabla, revisa que elementos no deberían ser nulos.
- Ahora, crea las tablas faltantes considerando la estructura que aquí se presenta, por el momento vamos a utilizar el modelo modificado la clase pasada donde colocamos los tipos.
- Ejecuta el commando `\dt` para listar todas las tablas creadas en Postgres



Práctica SQL

```
FOREIGN KEY clave_foranea REFERENCES tabla [(clave_primaria)]  
[ON DELETE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}]  
[ON UPDATE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}]
```

Debido a que las tablas quedan creadas pero no tiene relaciones, es necesario ejecutar este conjunto de comandos



Práctica SQL

```
ALTER TABLE public."PERFIL" ADD CONSTRAINT fk_cedula_usuario FOREIGN KEY  
(cedula)  
REFERENCES public."USUARIO" (cedula) MATCH FULL  
ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Debido a que las tablas quedan creadas pero no queda relacionadas, es necesario ejecutar este conjunto de comandos



Concordancia

Al momento de crear el constraint para las llaves foráneas nos podemos encontrar con:

MATCH FULL, no permitirá que una columna de una llave foránea de varias columnas sea nula, a menos que todas las llaves sean nulas.

MATCH SIMPLE, permite que cualquiera de las columnas de clave externa sea nula



Constraints

Los constraints en una base de datos corresponden a las restricciones que esta debe tener para que las reglas de negocio se cumplan. Estas restricciones están basadas en:

- Elementos no nulos
- Clave única
- Clave principal para identificar la tabla
- Clave foránea para relacionar tablas entre ellas
- Permisos y otras reglas aplicadas sobre las tablas
- Restricciones informativas



Práctica SQL

También es válido declarar la sentencia de la siguiente manera

```
ALTER TABLE public."PERFIL" ADD CONSTRAINT fk_cedula_usuario FOREIGN KEY  
(cedula)  
REFERENCES public."USUARIO" (cedula);
```

Realiza este proceso para las relaciones de todas las tablas



Práctica SQL

Nombre de la tabla

```
INSERT INTO productos  
VALUES (1250, 'LENA', 'Mesa', 'Diseño Juan Pi. Año 1920.', 25000);
```

Valores de la fila

El paso siguiente, cuando ya tenemos todas nuestras tablas creadas es insertar datos e información.



Práctica SQL

El orden de la información a guardar debe estar igual a como se creó en la tabla

```
INSERT INTO "USUARIO"  
VALUES(  
    cedula,  
    nombre,  
    apellido,  
    nombre_usuario,  
    correo,  
    clave,  
    activo);
```



Práctica SQL

Para insertar múltiples datos se puede utilizar la siguiente estructura

```
INSERT INTO "USUARIO" (cedula, nombre,  
apellido, nombre_usuario, correo, clave,  
activo)  
VALUES(  
    cedula,  
    nombre,  
    apellido,  
    nombre_usuario,  
    correo,  
    clave,  
    activo  
)  
(  
    cedula,  
    nombre,  
    apellido,  
    nombre_usuario,  
    correo,  
    clave,  
    activo  
);
```



Práctica SQL

Comandos muy utilizados de SQL que nos quedan pendientes por ver en clase:

- DROP TABLE
- ALTER TABLE
- CREATE SCHEMA
- GRANT
- UPDATE
- DELETE from
- CREATE view

Antes de dar inicio a las sentencias de consulta, debemos aprender sobre el proceso de normalización para garantizar que nuestros datos están limpios, óptimos y en buenas condiciones para crear consultas más rápidas.



REFERENCIAS

- Type of constraints: <https://www.ibm.com/docs/en/db2/10.5?topic=constraints-types>
- Constraints: <https://www.ibm.com/docs/es/db2-warehouse?topic=objects-constraints>
- El lenguaje SQL, Carme Martín Escofet

