DDL

Bases de Datos 1 (CC3088) - 2026

DDL – Data Definition Language

Semestre 01, 2026

Antes de consultar

Antes de escribir consultas necesitamos responder:

- ¿Quién define la estructura?
- ¿Qué impide guardar datos inválidos?
- ¿Cómo se relacionan las tablas? Sin estructura no hay base de datos confiable.

¿Qué ocurre sin reglas?

- Datos duplicados
- Valores inválidos
- Relaciones inconsistentes
- Errores difíciles de detectar Primero arquitectura. Después datos.

SQL

Structured Query Language.

Es el lenguaje estándar para bases de datos relacionales.

¿Qué permite hacer SQL?

- Definir estructura
- Insertar datos
- Consultar información
- Modificar registros
- Controlar acceso Hoy veremos la parte estructural.

Categorías

SQL se divide en grandes grupos:

- DDL – Data Definition Language
- DML – Data Manipulation Language
- DCL – Data Control Language
- TCL – Transaction Control Language Comenzamos con DDL.

DDL

DDL define la estructura.

No manipula datos.

Define qué es la base de datos.

DDL responde

- ¿Qué tablas existen?

- ¿Qué columnas tienen?
- ¿Qué tipos de datos aceptan?
- ¿Qué reglas deben cumplirse? DDL es arquitectura.

Tablas

Una tabla representa una entidad.

- Filas → registros
- Columnas → atributos
- Restricciones → reglas

Primero definimos estructura. Luego almacenamos datos.

Create

CREATE se usa para crear objetos.

```
CREATE TABLE estudiantes (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100)
);
```

Crea una tabla con identificador único y nombre.

- CREATE TABLE → instrucción principal
- id SERIAL → entero autoincremental
- PRIMARY KEY → identificador único
- VARCHAR(100) → texto con límite

Tipos de datos comunes

- INTEGER
- SERIAL
- VARCHAR(n)
- TEXT
- DATE
- NUMERIC
- BOOLEAN

Restricciones

Las restricciones garantizan integridad.

- PRIMARY KEY
- NOT NULL
- UNIQUE
- CHECK
- FOREIGN KEY

NOT NULL

Impide valores vacíos.

```
nombre VARCHAR(100) NOT NULL
```

Asegura que siempre exista un valor.

UNIQUE

Evita duplicados.

```
email VARCHAR(150) UNIQUE
```

Ideal para correos o identificadores externos.

CHECK

Define reglas lógicas.

```
presupuesto NUMERIC CHECK (presupuesto > 0)
```

Evita datos inválidos.

FOREIGN KEY

Define relaciones entre tablas.

```
CONSTRAINT fk_departamento
FOREIGN KEY (departamento_id)
REFERENCES departamentos(id)
```

Mantiene coherencia entre tablas.

Ejemplo

```
CREATE TABLE proyectos (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(150) NOT NULL,
    departamento_id INTEGER,
    fecha_inicio DATE DEFAULT CURRENT_DATE,
    presupuesto NUMERIC CHECK (presupuesto >= 1000),
    CONSTRAINT fk_departamento
        FOREIGN KEY (departamento_id)
        REFERENCES departamentos(id)
);
```

ALTER

ALTER modifica estructuras existentes. No destruye la tabla. La evoluciona.

Agregar columna

```
ALTER TABLE empleados
ADD COLUMN email VARCHAR(150);
```

Modificar tipo

```
ALTER TABLE empleados
ALTER COLUMN nombre TYPE VARCHAR(200);
```

Permite ajustes sin recrear la tabla.

Agregar restricción

```
ALTER TABLE empleados
ADD CONSTRAINT chk_email
CHECK (email LIKE '%@%');
```

Añade reglas después de creada la tabla.

DROP

DROP elimina objetos.

Es permanente.

```
DROP TABLE empleados;
```

Debe usarse con cuidado.

Variante segura

```
DROP TABLE IF EXISTS empleados;
```

Evita errores si la tabla no existe.

TRUNCATE

TRUNCATE elimina todos los registros. No elimina la estructura.

```
TRUNCATE TABLE empleados;
```

Es más rápido que DELETE. No permite filtros.