Módulo 4.2: Bases de datos Relacionales

MSc. Carlos Cedeño

Profundizando en Bases de Datos Relacionales (SQL)

Las bases de datos relacionales son el pilar del almacenamiento de datos estructurados desde hace décadas. Su modelo, basado en la lógica de predicados y la teoría de conjuntos, aporta orden, previsibilidad y, sobre todo, **integridad a los datos**.

1. El Modelo Relacional: Los Fundamentos Teóricos

El modelo fue propuesto por Edgar F. Codd en 1970. No se llama "relacional" por las "relaciones" entre tablas, sino porque su unidad fundamental es la **"relación"**, que en términos matemáticos es un conjunto de tuplas (filas).

- **Relación (Tabla):** Es una estructura de dos dimensiones compuesta por filas y columnas. En una base de datos de IoT, podrías tener una tabla `Dispositivos`.
- **Tupla (Fila o Registro):** Representa un único elemento de datos dentro de una tabla. Cada fila en la tabla `Dispositivos` sería un dispositivo específico (ej: "Sensor de Temperatura del Salón").
- **Atributo (Columna o Campo):** Define una propiedad de la tupla. Cada columna describe una característica del dispositivo, como `id_dispositivo`, `modelo`, `fecha_instalacion`, `ubicacion`.
- **Dominio:** Es el conjunto de valores permitidos para un atributo. Por ejemplo, el dominio del atributo `estado` podría ser `{'activo', 'inactivo', 'mantenimiento'}`. Esto asegura que no se puedan insertar valores inválidos.

2. Claves y Relaciones: Construyendo el Esqueleto

Las claves son las herramientas que permiten conectar los datos de forma lógica y sin ambigüedades.

- **Clave Primaria (Primary Key - PK):** Es un atributo que **identifica de forma única cada fila** en una tabla. No puede contener valores nulos y sus valores no deben repetirse. Por ejemplo, `id_dispositivo` en la tabla `Dispositivos`.
- **Clave Foránea (Foreign Key - FK):** Es un atributo en una tabla que apunta a la Clave Primaria de otra tabla. Es el **mecanismo para crear una relación**. Por ejemplo, si tenemos una tabla `Lecturas_Sensores`, podría tener una FK llamada `dispositivo_id` que apunta a la PK de la tabla `Dispositivos`. Así, sabemos qué dispositivo realizó cada lectura.

Estas claves permiten establecer los tres tipos de relaciones:

1. **Uno a Uno (1:1):** Una fila en la Tabla A se relaciona con una única fila en la Tabla B. (Ej: Un `Dispositivo` tiene una única `Configuracion_Avanzada`).
2. **Uno a Muchos (1:N):** Una fila en la Tabla A se relaciona con muchas filas en la Tabla B. **Esta es la relación más común**. (Ej: Un `Dispositivo` tiene muchas `Lecturas_Sensores`).
3. **Muchos a Muchos (N:M):** Muchas filas en la Tabla A se relacionan con muchas filas en la Tabla B. Se implementa usando una "tabla intermedia" o "tabla de unión". (Ej: Muchos `Usuarios` pueden tener acceso a muchos `Dispositivos`).

3. Normalización: El Arte de Organizar Datos

La normalización es el proceso de diseñar la estructura de la base de datos para **minimizar la redundancia de datos y evitar anomalías** (problemas al insertar, actualizar o borrar datos). Se logra dividiendo tablas grandes en tablas más pequeñas y bien estructuradas.

Link para aprender: https://www.youtube.com/watch?v=kvt2wE-q-yY

4. SQL (Structured Query Language): El Lenguaje para Hablar con los Datos

SQL es el lenguaje estándar para gestionar y consultar bases de datos relacionales. Se divide en sub-lenguajes:

DDL (Data Definition Language) - Para definir la estructura

- `CREATE TABLE` : Crea una nueva tabla.

```sql
CREATE TABLE Dispositivos (
    id_dispositivo INT PRIMARY KEY AUTO_INCREMENT,
    modelo VARCHAR(50) NOT NULL,
    ubicacion VARCHAR(100),
    fecha_instalacion DATE
);
```

- `ALTER TABLE` : Modifica una tabla existente (añade, elimina o modifica columnas).

```sql
ALTER TABLE Dispositivos ADD COLUMN estado VARCHAR(20) DEFAULT 'activo';
```

- `DROP TABLE` : Elimina una tabla completa.

DML (Data Manipulation Language) - Para manipular los datos

- `INSERT INTO` : Añade una nueva fila.

```
INSERT INTO Dispositivos (modelo, ubicacion, fecha_instalacion)
VALUES ('DHT22', 'Invernadero A', '2025-07-15');
```

- `UPDATE` : Modifica filas existentes.

```
UPDATE Dispositivos SET ubicacion = 'Invernadero B' WHERE id_dispositivo = 1;
```

- `DELETE FROM` : Elimina filas.

```
DELETE FROM Dispositivos WHERE id_dispositivo = 1;
```

DQL (Data Query Language) - Para consultar los datos

- `SELECT` : La instrucción más potente y usada. Recupera datos.

```sql
-- Seleccionar todos los dispositivos de un modelo específico
SELECT * FROM Dispositivos WHERE modelo = 'DHT22';

-- Contar cuántos dispositivos hay en cada ubicación
SELECT ubicacion, COUNT(id_dispositivo)
FROM Dispositivos
GROUP BY ubicacion;
```

- `JOIN` : **La operación relacional por excelencia.** Combina filas de dos o más tablas basándose en una columna relacionada.

```sql
-- Obtener las lecturas junto con la ubicación del dispositivo que las tomó
SELECT L.timestamp, L.valor, D.ubicacion
FROM Lecturas AS L
INNER JOIN Dispositivos AS D ON L.dispositivo_id = D.id_dispositivo
WHERE D.ubicacion = 'Invernadero A';
```

5. Transacciones y Propiedades ACID

Una **transacción** es una secuencia de operaciones que se ejecutan como un único bloque lógico de trabajo. O se completan todas con éxito, o no se realiza ninguna. Esto garantiza la integridad de los datos, lo cual es vital.

Las transacciones en bases de datos relacionales siguen las propiedades **ACID**:

- **A - Atomicidad (Atomicity):** Una transacción es "todo o nada". Si una parte falla, toda la transacción se revierte (`ROLLBACK`).
    - **Ejemplo IoT:** Al dar de alta un dispositivo, necesitas `INSERTAR` en la tabla `Dispositivos` y también en la tabla `Configuraciones` . Si la segunda inserción falla, la primera debe deshacerse para no dejar un dispositivo sin configuración.
- **C - Consistencia (Consistency):** La base de datos siempre pasa de un estado válido a otro estado válido. Ninguna transacción puede dejar los datos en un estado inconsistente (ej: violando una clave foránea o una restricción).

- **I - Aislamiento (Isolation):** Las transacciones concurrentes (que ocurren al mismo tiempo) no deben interferir entre sí. El resultado debe ser el mismo que si se hubieran ejecutado una tras otra.
- **D - Durabilidad (Durability):** Una vez que una transacción ha sido confirmada (`COMMIT`), los cambios son permanentes y sobrevivirán a cualquier fallo del sistema (como un corte de energía).