ChatGPT

Lenguaje DDL y definición de tablas

El **Lenguaje de Definición de Datos (DDL)** es la parte de SQL usada para crear y modificar la estructura de la base de datos [1]. A través de DDL se definen tablas, campos, tipos de datos y restricciones (como *PRIMARY KEY* o *FOREIGN KEY*) que garantizan la integridad de los datos [1] [2]. Sus comandos principales son **CREATE** (crea objetos), **ALTER** (modifica estructuras) y **DROP** (elimina objetos) [2] [3]. Por ejemplo, `CREATE TABLE empleados (id INT PRIMARY KEY, nombre VARCHAR(50))` genera una nueva tabla **empleados** con campos `id` y `nombre` [4] [5]. DDL no gestiona los datos en sí (eso corresponde al lenguaje DML), sino el **esquema** de la base de datos [1].

- **CREATE TABLE**: crea una tabla nueva con los campos y restricciones definidas. Sintaxis básica:

```
CREATE TABLE nombre_tabla (
    campo1 TIPO [restricciones],
    campo2 TIPO [restricciones],
    ...
);
```

  Cada columna se define por su nombre, tipo de dato y restricciones opcionales [5].
- **Tipos de datos y nulidad**: al crear campos se especifica el tipo de dato (INT, VARCHAR, DATE, etc.) y si admiten **NULL** o no. La restricción `NOT NULL` impide valores nulos en esa columna [6]. Por ejemplo: `nombre VARCHAR(100) NOT NULL` asegura que cada registro tenga un valor en *nombre*.
- **Llave primaria (PRIMARY KEY)**: identifica de forma única cada fila. Se define con `PRIMARY KEY` sobre uno o varios campos [7] [8]. Una tabla sólo puede tener una clave primaria; en ella no puede repetirse ningún valor [8]. Además, por estándar SQL las columnas de la clave primaria son implícitamente `NOT NULL` [9].
- **Llave foránea (FOREIGN KEY)**: enlaza dos tablas, asegurando *integridad referencial*. Se declara con `FOREIGN KEY (campo)` que referencia la clave primaria de otra tabla [10] [11]. Esto obliga

a que todo valor de la clave foránea exista previamente como clave primaria en la tabla referenciada.

Ejemplo de creación de tablas

Con DDL se puede implementar un modelo relacional concreto. Por ejemplo, para una base de datos escolar podríamos crear:

```
CREATE TABLE Departamento (
    depto_id INT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL
);

CREATE TABLE Empleado (
    id INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    salario DECIMAL(10,2),
    depto_id INT,
    FOREIGN KEY (depto_id) REFERENCES Departamento(depto_id)
);
```

Este código crea la tabla **Departamento** con clave primaria `depto_id`, y la tabla **Empleado** con clave primaria `id` y llave foránea `depto_id` que refiere a `Departamento.depto_id` [10] [12]. Los tipos de dato (INT, VARCHAR, DECIMAL, DATE, etc.) y las restricciones (*PRIMARY KEY*, *NOT NULL*, *FOREIGN KEY*) garantizan que la estructura refleje fielmente el modelo de datos.

En un **modelo entidad-relación (ER)** como el de la imagen se ilustra cómo las tablas (entidades) se conectan mediante claves. La integridad referencial exige que cada campo marcado como *FOREIGN KEY* contenga un valor existente en la tabla padre [11] [12]. Por ejemplo, la restricción `FOREIGN KEY (depto_id) REFERENCES Departamento(depto_id)` impide insertar un empleado con `depto_id` inexistente, manteniendo las relaciones sincronizadas [12] [11]. En resumen, DDL permite **implementar modelos de datos relacionales** con todas sus dependencias: definiendo llaves primarias, foráneas y otras reglas de integridad para asegurar la coherencia de la base de datos [11] [12].

Modificación de tablas con ALTER TABLE

Las tablas existentes pueden alterarse con **ALTER TABLE**. Este comando permite **añadir, eliminar o modificar columnas y restricciones** manteniendo los datos existentes [13]. Por ejemplo, para agregar un campo fecha de nacimiento a *Empleado* se usaría:

```
ALTER TABLE Empleado
ADD COLUMN fecha_nac DATE;
```

O para cambiar el tipo de dato de `salario`:

```
ALTER TABLE Empleado
MODIFY COLUMN salario DECIMAL(12,2);
```

Estos ejemplos añaden una columna y modifican otra [14] [15]. También se puede renombrar columnas (`CHANGE COLUMN viejo_nombre nuevo_nombre TIPO`) o tablas (`RENAME TO`), y establecer valores por defecto o restricciones adicionales [16] [17].

Para ajustar **nulidad** de una columna (por ejemplo, hacer que admita NULL o no) se suele usar `ALTER TABLE nombre_tabla MODIFY columna TIPO [NOT NULL]` en MySQL o sintaxis equivalente según SGBD [6] [15]. Es vital tener cuidado con ALTER en tablas grandes, ya que puede bloquearlas; siempre se recomienda probar cambios primero y garantizar respaldos [18].

Eliminación y truncado de tablas

Finalmente, DDL incluye comandos para eliminar datos o estructuras completas. El comando **TRUNCATE TABLE** borra **todos los registros** de una tabla, pero conserva su definición (esquema) [19]. Es más rápido que un `DELETE`, pues no recorre fila a fila y suele restablecer contadores *AUTO_INCREMENT*. Por ejemplo, `TRUNCATE TABLE Empleado;` vacía la tabla *Empleado* sin borrarla [19].

En cambio, **DROP TABLE** elimina **por completo** la tabla, su estructura, datos y dependencias [20]. Por ejemplo, `DROP TABLE Empleado;` quita *Empleado* de la base de datos definitivamente [20]. No debe usarse a la ligera, ya que borra índices, claves y disparadores asociados. En resumen:
- `DELETE` elimina filas específicas (usando *WHERE*),
- `TRUNCATE TABLE` elimina rápidamente *todas* las filas (DDL, conserva esquema) [21],
- `DROP TABLE` elimina la tabla entera y su definición [20].

Fuentes: Conceptos y ejemplos de DDL (CREATE TABLE, ALTER TABLE, DROP TABLE, TRUNCATE TABLE) obtenidos de documentación SQL y tutoriales especializados [1] [5] [13] [20]. Se han citado referencias en español para definiciones de restricciones *NOT NULL* [6], claves primarias [8], y la integridad referencial [11] [12].

---

[1] [2] [4] [12] DDL: Lenguaje de Definición de Datos | Centro de Conocimiento
https://www.datasunrise.com/es/centro-de-conocimiento/ddl-lenguaje-de-definicion-de-datos/

[3] [19] [20] [21] TRUNCATE TABLE vs. DELETE vs. DROP TABLE: Eliminación de tablas y datos en SQL | LearnSQL.es
https://learnsql.es/blog/truncate-table-vs-delete-vs-drop-table-eliminacion-de-tablas-y-datos-en-sql/

[5] [7] [10] Declaración MySQL CREATE TABLE: Uso y ejemplos
https://www.datacamp.com/es/doc/mysql/mysql-create-table

[6] Restricciones NOT NULL
https://www.ibm.com/docs/es/db2/11.1.0?topic=constraints-not-null

[8] [9] Clave primaria - Wikipedia, la enciclopedia libre
https://es.wikipedia.org/wiki/Clave_primaria

[11] Integridad referencial - Wikipedia, la enciclopedia libre
https://es.wikipedia.org/wiki/Integridad_referencial

13 14 15 16 17 18 Declaración ALTER TABLE de MySQL: Uso y ejemplos
https://www.datacamp.com/es/doc/mysql/mysql-alter-table