Consultas SQL 3

IT BOARDING

BOOTCAMP

En este módulo vamos a explicar los conceptos teóricos y prácticos referidos a las sentencias particulares y objetos fundamentales dentro de una base de datos.

IT BOARDING

BOOTCAMP

Índice

IT BOARDING BOOTCAMP

Normalización de una base de datos

La normalización es un proceso de **estandarización** y **validación** de **datos** que consiste en eliminar las **redundancias** o **inconsistencias**, completando datos mediante una serie de reglas que actualizan la información, protegiendo su **integridad** y **favoreciendo** la **interpretación**, para que así sea más simple de **consultar** y más eficiente para quien la **gestiona**.

Normalización de una base de datos

NIVELES DE NORMALIZACIÓN

1NF: Elimina datos duplicados en atributos. Crea registros independientes.

(EJ: SIN Atributos con múltiples valores, Tabla Persona, Columna Teléfono, 2 teléfonos en el campo)

2NF: Elimina columnas que no dependen de la clave principal.

(Ej: SIN Dependencias Parciales, Tabla Alumnos, Columna Id, Columna Curso, Columna Valor del curso)

3NF: Elimina subgrupos de datos en múltiples columnas de una tabla y crea tablas nuevas, con relaciones entre ellas.

(Ej: SIN Dependencias Transitivas, tabla Alumnos, Columna Id, Columna Provincia depende de Id, Columna Localidad depende de Provincia, entonces Columna Localidad depende de Id del Alumno)

4NF: Desaparecen todas las dependencias.

Normalización de una base de datos

SIN NORMALIZAR

- DATOS **ERRÓNEOS**
- DATOS **REDUNDANTES**
- DATOS **DESACTUALIZADOS**
- DATOS **INSUFICIENTES**
- DATOS **NULOS**
- DATOS **MAL TIPIFICADOS**

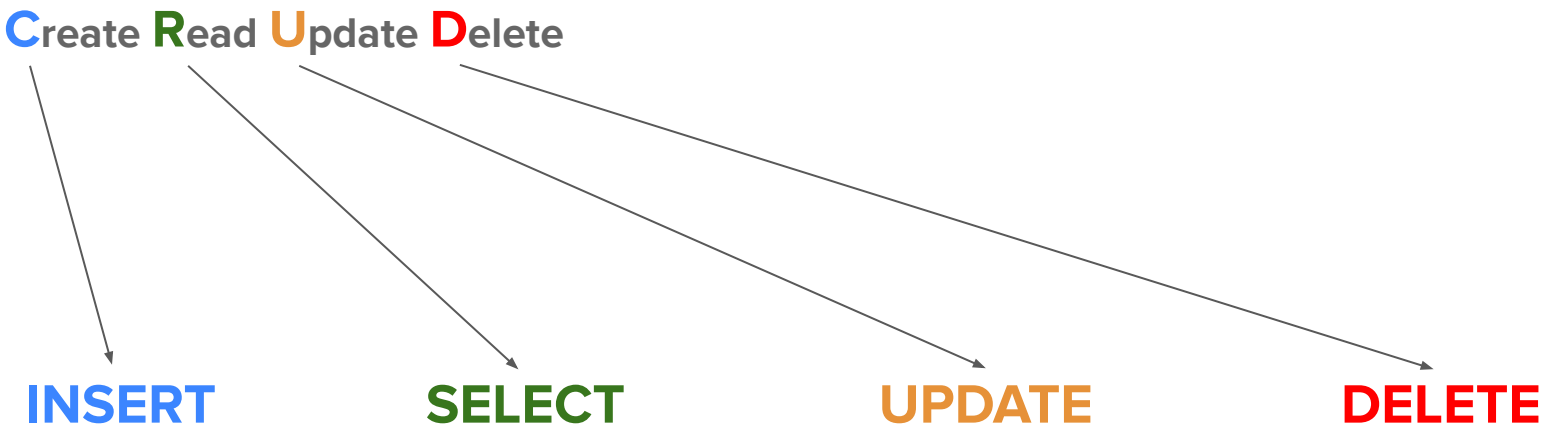
NORMALIZADA

- DATOS **PRECISOS**
- DATOS **ÚNICOS**
- DATOS **ÍNTEGROS**
- DATOS **COMPLETOS**
- DATOS **RELEVANTES**
- DATOS **BIEN TIPIFICADOS**

Sentencias DML

(data manipulation language)

Son aquellas utilizadas para **insertar**, **consultar**, **modificar** o **borrar** los **datos** de una base de datos.



Insert

SINTAXIS

```
INSERT INTO <table_name> (column1, column2, ..., columnN)
SELECT column1, column2, …, columnN
FROM  <source table_name>;

INSERT INTO <table_name>(column1,column2, …, columnN)
VALUES (value1, value2, …, valueN);
```

REGLAS

- El número de columnas especificado en la lista VALUES debe coincidir con las columnas especificadas en la cláusula **INSERT INTO**.
- Los valores son obligatorios para las columnas **NOT NULL**.
- Si no se especifican valores, se inserta **NULL** para los campos nullables.
- Los tipos de datos de las columnas especificadas en la cláusula **VALUES** deben ser compatibles con los tipos de datos de las columnas en la cláusula **INSERT**.

Insert

SINTAXIS + EJEMPLO

```
INSERT INTO actors
(first_name, last_name, rating, favorite_movie_id)
VALUES
('Charles','Creek', 9.0, 11);
```

Update

SINTAXIS

```
UPDATE <table_name>
SET   <column1> = <new_value1>
      <column2> = <new_value2>
                  ....
      <columnN> = <new_valueN>
[WHERE  condition];
```

REGLAS

- Puede actualizar uno o más campos de la tabla.
- Si no se especifica la condición WHERE, todas las filas de la tabla se van a actualizar.
- Puede actualizar una tabla con los valores de otra tabla.

Update

SINTAXIS + EJEMPLO

```
UPDATE actors
SET rating = 8.0,
    favorite_movie_id = 2
WHERE last_name = ‘Creek’;
```

Delete

SINTAXIS

```
DELETE FROM <tablename> [WHERE <condition>];
```

REGLAS

- Puede borrar uno o más registros de la tabla.
- **Si no se especifica la condición WHERE, se eliminan todas las filas de la tabla.**
- Puede borrar una tabla a partir de los valores de otra tabla.

Delete

SINTAXIS + EJEMPLO

```
DELETE FROM actors
WHERE last_name = ‘Creek’;
```

Tablas temporales

¿QUÉ SON Y PARA QUÉ SE UTILIZAN?

- Son **tablas temporales** que generalmente las utilizamos para hacer pruebas, consultas, análisis, cargas en tablas de staging, etc.
- Se pueden **realizar operaciones** de select, insert, delete, update.
- La tabla y sus datos se eliminan al finalizar la sesión.
- Para **evitar el uso de múltiples joins** en una misma consulta.

Tablas temporales

¿QUÉ SON Y PARA QUÉ SE UTILIZAN?

ALGUNAS CARACTERÍSTICAS

- **No se pueden compartir a otros usuarios.**
- Se pueden crear sin necesidad de permisos especiales.
- Se pueden crear hasta 1000 tablas volátiles en una sesión.
- **Los datos no quedan guardados de manera permanente.**

Tablas temporales

¿QUÉ SON Y PARA QUÉ SE UTILIZAN?

SINTAXIS DE CREACIÓN

```
CREATE TEMPORARY TABLE <table_name>
            <query para obtener los datos a insertar>;
```

```
CREATE TEMPORARY TABLE <table_name>
                (column1 dataType [not null] [primary key],
                 column2 dataType [not null] [primary key],
                 …,
                 columnN dataType [not null] [primary key]);
```

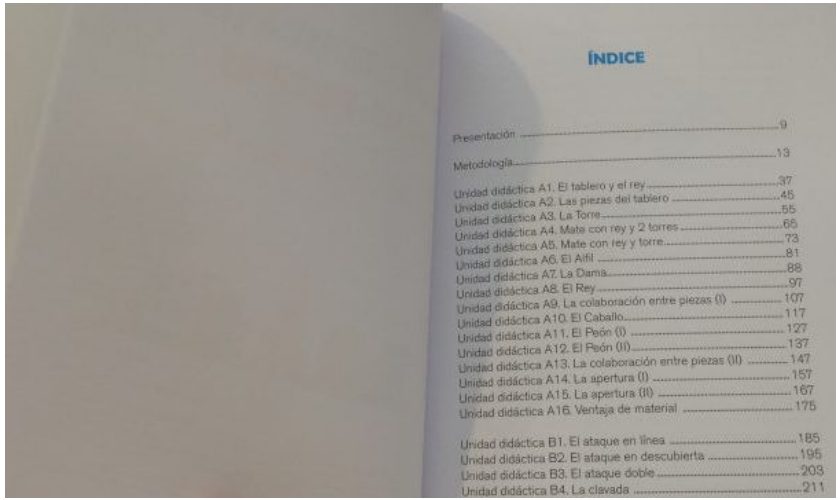
EXPLAIN PLAN

(Plan de ejecución)

- Un plan de ejecución define la forma en que la base busca o graba los datos.
- Decide si la consulta va a usar o no los **índices** en una sentencia SELECT.
- Indica si una instrucción requerirá “barrer” una tabla completa.
- Indica el tiempo de ejecución estimado.

Cada base de datos posee una sintaxis y forma de ejecutar o mostrar los planes de ejecución.

índices



ÍNDICE

Índices

¿QUÉ SON Y PARA QUÉ SIRVEN?

- Son un mecanismo para **optimizar** consultas en SQL.
- Mejoran sustancialmente los tiempos de respuesta en Queries complejas.
- **Mejoran el acceso a los datos** al proporcionar una ruta más directa a los registros.
- Evitan realizar escaneos (barridas) completas o lineales de los datos en una tabla.

Índices

TIPOS DE ÍNDICE

- **Índice de clave primaria**
  - No admite duplicados
- **Índice ordinario**
  - Admite duplicados
- **Índice único**
  - Son como los índices ordinarios pero no admiten duplicados

Índices

RECOMENDACIONES PARA DETERMINAR UN ÍNDICE

- **Comprender cabalmente** tanto la **base de datos** como el **negocio.**
- Entender los “**WHERE**” y los “**JOIN**” de las consultas que se realizan para poder “predecir” próximas consultas recurrentes.
- Es habitual seleccionar como **Primary Index** el **id** de la tabla así como también **fechas** o id de País/Región.
- **Puede involucrar uno o más campos**.
- Seleccionar columnas que no cambien regularmente.

Índices

ALGUNAS FORMAS PARA DETECTAR UN ÍNDICE DE UNA TABLA

1. **SHOW INDEX FROM <table_name>;**
2. Validar el **índice** de la tabla.
3. HELP INDEX <schema.table_name>;

Índices

SINTAXIS PARA CREAR UN ÍNDICE EN SCRIPT DE CREACIÓN DE TABLA

```
CREATE TABLE Tabla(
        Columna1 int,
        Columna2 varchar(255),
        ...
        ColumnaN DATA_TYPE
)
UNIQUE
PRIMARY INDEX(Columna1, ...,ColumnaN);
```

```
CREATE TABLE Tabla(
        Columna1 int,
        Columna2 varchar(255),
        ...
        ColumnaN DATA_TYPE
)

PRIMARY INDEX(Columna1, ..., ColumnaN);
```

Índices

SINTAXIS

```
CREATE [UNIQUE] INDEX <index_name>
ON <tablename> (column1, column2, …, columnN);

ALTER TABLE <table_name>
ADD INDEX nombre_indice(column1, column2, …, columnN);
```

EJEMPLO

```
CREATE INDEX movies_idx
    ON MOVIES (id);
```

Gracias.

IT BOARDING

BOOTCAMP