**Actividad 1: Llaves Primarias y Foráneas**

1. **Llave Primaria**

Una llave primaria es un identificador púnico para cada registro ingresado a una tabla. Su función principal es hacer que cada fila en la tabla se pueda identificar. Las llaves primarias deben cumplir con las condiciones de no pueden tener valores repetidos ni nulos.

1. **Llave Foránea**

Es una columna o un conjunto de columnas en una tabla cuyos valores corresponden a los valores de la clave primaria de otra tabla.

**• Relación entre llaves primarias y llaves foráneas / Importancia para establecer relaciones entre tablas**: Las llaves primarias y las llaves foráneas son esenciales en bases de datos relacionales para establecer y mantener relaciones entre tablas. Esta conexión crea vínculos entre tablas y asegura que los datos sean consistentes y no se creen relaciones inválidas entre registros.

**• Utilidad de las llaves foráneas:**

Las llaves foráneas conectan tablas. Esto ayudará a las consultas que pueden extraer y combinar datos relacionados de diferentes tablas de una manera eficiente. Esto hará que la base de datos mantenga la consistencia y la integridad de los datos.

**Actividad 2: ¿Qué es SQL?**

**Definición de SQL**

**SQL** o **Structured Query Language**. Es un **Lenguaje de Consulta Estructurado**. Este tipo de **lenguaje de programación** permite manipular y descargar datos de una base de datos. Es capaz de hacer cálculos avanzados y álgebra. Además, es usado en la mayoría de empresas que almacenan datos en una base de datos.

**Conceptos Clave**:

* **Consulta (Query)**: Una solicitud para obtener datos específicos de una base de datos.
* **Transacción (Transaction)**: Un conjunto de operaciones que se ejecutan como una unidad indivisible.
* **Tabla (Table)**: Una estructura que organiza los datos en filas y columnas.
* **Columna (Column)**: Un campo dentro de una tabla que almacena datos de un tipo específico.
* **Registro (Record)**: Una fila en una tabla que contiene datos.

**Importancia de interactuar con bases de datos usando SQL:**

Saber cómo interactuar con estos lenguajes es importante, porque permite a los usuarios extraer, manipular y gestionar datos de manera eficiente. Esta es una habilidad básicamente necesaria y fundamental para un desarrollador o profesional que trabaje con datos.

**Término que más me interesó y por qué**:

* **Transacción**: Ya que asegura que un conjunto de operaciones se ejecute de manera unida y consistente. Esto es crítico en, por ejemplo, en aplicaciones de sectores financieros y sistemas donde la integridad de los datos es fundamental. Esto significa que una transacción debe completarse en su totalidad o no ejecutarse en absoluto, lo cual asegura que los datos no queden en un estado parcial o corrupto si algo falla durante el proceso. Un ejemplo de esto es una transferencia de fondos, donde se debe deducir un monto de una cuenta y sumarlo en otra.

**Actividad 3: Tipos de JOINS y sus usos.**

El Join es sin duda una de las operaciones más importantes en SQL pero también una de las más difíciles de entender. Para ilustrar los diferentes tipos de Join, la imagen que encabeza este post ofrece un resumen (utilizando teoría de conjuntos) de cómo se comporta cada Join. Pasemos a explicar con un poco más de detalle cada uno de ellos:

* **Inner Join. El Join de “toda la vida”.** El join per defecto que se aplica cuando no indicamos otra cosa al hacer la consulta. Devuelve sólo aquellas filas donde haya un “match”, es decir, las filas donde el valor del campo de la tabla A que se utiliza para hacer el Join coincida con el valor del campo correspondiente en la tabla B.  Ejemplo: devolver todos los productos para los que haya como mínimo un pedido en los últimos días (el inner join enlazará el campo producto en la tabla Pedido con la clave primaria de ese producto en la tabla Producto).
* **Left outer Join.**Cuando quieres todas las filas para las que haya match pero también aquellas de la Tabla A que no hagan match. Siguiendo el ejemplo anterior, si quieres listar todos los productos con datos de sus pedidos pero mostrando también aquellos productos para lo que no tengas todavía un pedido, la solución sería hacer una Left Outer join entre Producto y Pedido.
* **Right outer Join.** Exactamente lo mismo pero a la inversa, cuando quieres listar las filas de la tabla B aunque no estén relacionadas con ninguna fila de la tabla A. Es un operador un poco redundante ya que se podría cambiar simplemente el orden de las  tablas en el Join y utilizar un left outer para conseguir el mismo efecto. No obstante, y como parte de Joins múltiples, es útil tener los dos para una mejor comprensión de la consulta
* **Full outer join**. Es como la suma de las dos anteriores. Queremos tanto las filas de la A como las de B, tanto si hay match como si no (evidentemente cuando haya match la consulta devolverá todos los campos de A y B que hayamos indicado, cuando no, la consulta devolverá sólo los campos de A o B).

**Actividad 4: 8 Comandos útiles de SQL:**

* 1. **DISTINCT:** Evita valores duplicados en los resultados. Es útil para obtener listas únicas, por ejemplo, de categorías de productos.

**SELECT DISTINCT categoria FROM productos;**

* 1. **LIMIT:** Restringe la cantidad de filas devueltas. Útil para mostrar solo una cantidad específica de resultados, como los 5 primeros.

**SELECT \* FROM productos LIMIT 5;**

* 1. **CASE: Permite establecer condiciones dentro de una consulta. Es útil para crear valores calculados basados en condiciones.**

**SELECT nombre,**

**CASE**

**WHEN edad < 18 THEN 'Menor de edad'**

**ELSE 'Adulto'**

**END AS categoria\_edad**

**FROM clientes;**

* 1. **COALESCE:** Devuelve el primer valor no nulo de una lista. Es útil para manejar NULL y proporcionar valores predeterminados.

**SELECT nombre, COALESCE(telefono, 'No disponible') AS telefono FROM clientes;**

* 1. **UNION**: Combina los resultados de dos consultas eliminando duplicados. Es útil para unificar resultados de tablas similares.

**SELECT nombre FROM clientes**

**UNION**

**SELECT nombre FROM empleados;**

* 1. **GROUP BY:** Agrupa resultados según una o más columnas. Es útil para calcular agregados como sumas y promedios por grupo.

**SELECT categoria, COUNT(\*) FROM productos GROUP BY categoria;**

* 1. **HAVING:** Filtra los resultados de grupos, usado con GROUP BY. Es útil cuando quieres aplicar condiciones a agregaciones.

**SELECT categoria, COUNT(\*)**

**FROM productos**

**GROUP BY categoria**

**HAVING COUNT(\*) > 5;**

* 1. **IS NULL / IS NOT NULL:** Evalúa si un valor es NULL o no. Es útil para verificar y filtrar datos nulos.

**SELECT nombre FROM clientes WHERE telefono IS NULL;**