UNIDAD 1 - Bases de Datos Relacionales

Introducción a Bases de Datos

¿Qué es una Base de Datos?

Una **base de datos** es un conjunto organizado de información, habitualmente relacionado a un contexto específico, que se almacena de manera sistemática para su uso futuro. Esta estructura sistemática permite la fácil recuperación, gestión y manipulación de los datos cuando sea necesario.

La analogía más sencilla para entender una base de datos es imaginarla como una biblioteca organizada. Al igual que una biblioteca almacena libros de forma ordenada, una base de datos almacena información para facilitar su acceso y uso.

En el ámbito digital, las bases de datos son fundamentales para el funcionamiento de prácticamente todas las aplicaciones y sistemas, ya que permiten que los datos sean almacenados, actualizados y recuperados eficientemente.

Almacenamiento Sistemático de Datos

Los datos dentro de una base de datos se almacenan de manera que permiten el acceso sistemático y eficiente. Este almacenamiento puede ser comparado con el sistema de almacenamiento analógico, como el de una biblioteca o un archivero, donde la información está organizada de manera sistemática para facilitar el acceso.

En el mundo digital, los medios de almacenamiento como discos duros y discos de estado sólido son considerados sistemas de almacenamiento de tipo base de datos debido a cómo estructuran la información. Las aplicaciones de software de bases de datos están diseñadas específicamente para estructurar la información de tal manera que permita el acceso homogéneo y que los datos sean visualizados de la manera más clara posible.

Ejemplo de Estructura de Base de Datos

Consideremos un escenario en el que se necesita almacenar información de clientes y sus compras. En una base de datos plana, podríamos tener una única tabla donde cada fila

contiene toda la información del cliente y los detalles de la compra, repitiendo así la información del cliente para cada compra. Sin embargo, esto no es eficiente ya que genera redundancia de datos.

Por otro lado, en una base de datos relacional, la información se normaliza y se divide en múltiples tablas, como una tabla para clientes y otra para compras. Cada tabla tiene un propósito específico, y las relaciones entre estas tablas se establecen a través de claves primarias y foráneas. Esto no solo optimiza el espacio de almacenamiento, sino que también reduce la complejidad y la redundancia de datos.

Importancia de las Bases de Datos

Las bases de datos son vitales en diversos campos y aplicaciones, desde sistemas de gestión empresarial como ERP y CRM hasta servicios web y redes sociales. Su importancia radica en su capacidad para:

- **Evitar la duplicación de datos:** Mediante el uso de bases de datos relacionales, es posible reducir la redundancia al almacenar datos de manera normalizada.
- **Optimizar el espacio de almacenamiento:** Al estructurar la información correctamente, se utiliza el espacio de almacenamiento de manera más eficiente.
- **Facilitar el acceso y la manipulación de datos:** Gracias al uso de lenguajes de consulta como SQL, se puede acceder y manipular datos de manera eficaz, lo que es crucial para la toma de decisiones empresariales y el análisis de datos.

Las bases de datos continúan evolucionando, adaptándose a nuevas tecnologías y necesidades, como el análisis de grandes volúmenes de datos (big data), el aprendizaje automático (machine learning), y el Internet de las Cosas (IoT). Esta evolución constante asegura que seguirán siendo una herramienta indispensable en el mundo digital.

Componentes de una Base de Datos

Las bases de datos son estructuras diseñadas para almacenar y organizar datos de manera eficiente. Entender sus componentes principales, como tablas, registros y campos, es esencial para manejar y recuperar información en un sistema relacional.

Tablas

Las **tablas** son la estructura fundamental en una base de datos relacional. Similar a una hoja de cálculo, las tablas organizan los datos en filas y columnas. Cada tabla se enfoca en un tema o entidad específica, como clientes o productos.

- **Columnas (Campos)**: Representan las características de la entidad descrita por la tabla. Cada columna tiene un nombre único y un tipo de dato, como texto o números. Por ejemplo, en una tabla de clientes, las columnas pueden ser nombre, dirección y teléfono.

- **Filas (Registros)**: Contienen datos individuales de la tabla. Cada fila representa un registro único. Por ejemplo, en una tabla de productos, una fila podría incluir el nombre del producto, el precio y la cantidad en inventario.

Esquema de una Tabla

El **esquema de una tabla** define su estructura y especifica qué columnas contiene y qué tipo de datos se pueden almacenar. Esto garantiza que los datos ingresados sean consistentes y correctos.

Registros

Un **registro** es un conjunto de datos relacionados en una fila de una tabla. Representa una instancia de la entidad descrita. Por ejemplo, un registro en una tabla de ventas podría incluir detalles como el número de la transacción, el cliente y el monto total.

Clave Primaria

Para identificar cada registro de manera única, se utiliza una **clave primaria**. Es un campo con valores únicos para cada registro, asegurando que cada fila sea individualmente identificable.

Campos

Los **campos** son las columnas de una tabla y contienen tipos de datos específicos. Estos tipos pueden ser:

- **Texto**: Cadenas de caracteres.
- **Números enteros**: Números sin decimales.
- **Números decimales**: Números con decimales.
- **Fechas**: Valores de fecha y hora.

Propiedades de los Campos

Los campos pueden tener propiedades adicionales que aseguran la integridad de los datos, como restricciones de unicidad y valores predeterminados. Por ejemplo, una clave primaria debe ser única y no nula, garantizando que cada registro tenga un identificador exclusivo.

Importancia de los Componentes

Comprender estos componentes es esencial para un diseño eficaz de bases de datos, lo que resulta en:

- **Eficiencia en el almacenamiento**: Minimiza la redundancia de datos.
- **Integridad de los datos**: Asegura que los datos sean precisos y consistentes.
- **Facilidad de acceso**: Permite recuperar y manipular datos eficientemente utilizando SQL.

Diseñar correctamente las tablas, registros y campos es fundamental para crear bases de datos que puedan adaptarse y escalar con las necesidades de una organización.

Bases de Datos Relacionales

Introducción a Bases de Datos Relacionales

Las **bases de datos relacionales** son un tipo de sistema de gestión de bases de datos (DBMS) que utiliza un modelo basado en tablas para representar y gestionar datos. Este enfoque se fundamenta en el **modelo relacional**, desarrollado por E.F. Codd en 1970, el cual utiliza la lógica matemática para estructurar datos y relaciones.

Estructura Basada en Tablas

Las bases de datos relacionales están organizadas en tablas, donde cada tabla representa una entidad distinta. Cada fila en una tabla, conocida como **registro** o **tupla**, representa un elemento individual de esa entidad, mientras que cada columna, llamada **campo** o **atributo**, representa una propiedad del elemento. Esta estructura tabular permite que las tablas se relacionen entre sí, creando conexiones lógicas entre diferentes conjuntos de datos.

Tablas y Relaciones

- **Tablas**: Son el núcleo de las bases de datos relacionales. Cada tabla tiene un nombre único y consta de columnas y filas. Las columnas tienen un nombre y un tipo de dato específico (por ejemplo, texto, entero, fecha), y las filas contienen los datos de las entidades.

- **Relaciones**: Las bases de datos relacionales obtienen su nombre por la capacidad de definir relaciones entre tablas. Las relaciones se establecen utilizando claves:

    - **Clave Primaria (PK)**: La **llave primaria** o *primary key* es un identificador único para cada registro dentro de una tabla. Este campo o conjunto de campos garantiza que cada fila de la tabla pueda ser identificada de manera unívoca, evitando duplicados y asegurando que los datos sean únicos y no nulos. La clave primaria es fundamental para la estructura de las bases de datos relacionales, ya que permite mantener la integridad de los datos.

    Por ejemplo, en una tabla de clientes, el número de identificación o DNI podría servir como llave primaria, ya que no existen dos clientes con el mismo número. Esta llave no solo facilita la búsqueda y clasificación de los datos, sino que también permite la relación entre tablas al servir como punto de referencia.

    - **Clave Foránea (FK)**: La **llave foránea** o *foreign key* es un campo o conjunto de campos en una tabla que se utiliza para establecer y reforzar un vínculo entre los datos de dos tablas. Este tipo de llave apunta a una llave primaria en otra tabla, permitiendo que las bases de datos se conecten y compartan información de manera lógica y ordenada.

La llave foránea garantiza que la relación entre las tablas mantenga la integridad referencial. Esto significa que, por ejemplo, si una tabla de "Pedidos" tiene una llave foránea que apunta a una llave primaria en una tabla de "Clientes", cada pedido debe estar asociado a un cliente existente. Si un cliente es eliminado de la base de datos, las restricciones de integridad referencial evitarán que existan pedidos huérfanos sin un cliente asociado.

- **Clave Única (UK)**: La **llave única** o *unique key* es un campo o un conjunto de campos cuyos valores son únicos para cada registro dentro de una tabla, pero que no identifica a la misma ante las demás tablas. Ejemplo de esto podría ser el mail en una tabla de usuarios, cuya clave primaria es el Id de Usuario pero en el modelo se determina que el mail no se puede repetir.

Ventajas del Modelo Relacional

1. **Normalización**: Ayuda a minimizar la redundancia de datos y asegura la coherencia de la base de datos. La normalización divide los datos en tablas adicionales para reducir la duplicación y mejorar la integridad.
2. **Consultas Eficientes**: Las bases de datos relacionales utilizan SQL (Structured Query Language) para realizar consultas complejas y obtener reportes de manera eficiente. SQL es un lenguaje poderoso que permite recuperar, insertar, actualizar y eliminar datos de manera eficaz.
3. **Integridad Referencial**: Asegura que las relaciones entre tablas permanezcan consistentes. Por ejemplo, una clave foránea debe corresponder siempre a una clave primaria válida en la tabla relacionada, evitando así enlaces rotos y datos huérfanos.
4. **Escalabilidad y Flexibilidad**: Las bases de datos relacionales son adecuadas para manejar grandes volúmenes de datos y pueden adaptarse a nuevas necesidades de almacenamiento y procesamiento mediante la adición de más tablas y relaciones.

Tipos de Relaciones en Bases de Datos Relacionales

- **Uno a Uno (1:1)**: Cada fila en una tabla se relaciona con una única fila en otra tabla. Por ejemplo, un empleado puede tener un único número de identificación.
- **Uno a Muchos (1:n)**: Una fila en una tabla se puede relacionar con muchas filas en otra tabla. Por ejemplo, un departamento puede tener muchos empleados.
- **Muchos a Muchos (n:m)**: Muchas filas en una tabla pueden relacionarse con muchas filas en otra tabla, generalmente implementado mediante una tabla intermedia. Por ejemplo, estudiantes y cursos donde un estudiante puede inscribirse en múltiples cursos y un curso puede tener múltiples estudiantes.

Las bases de datos relacionales son una solución versátil y poderosa para la gestión de datos estructurados. Su capacidad para manejar grandes cantidades de información con eficiencia y precisión las hace esenciales en diversos sectores, desde la administración empresarial hasta la tecnología informática .

Aplicaciones Prácticas

Creación de una Base de Datos Simple

Instrucciones para la Creación de una Base de Datos Relacional Básica

Crear una base de datos relacional básica implica una serie de pasos cuidadosamente planificados que permiten establecer una estructura eficiente para el almacenamiento y manejo de datos. A continuación, se presenta una guía paso a paso que destaca la importancia de las relaciones y la normalización en este tipo de bases de datos.

Paso 1: Definición del Propósito de la Base de Datos

Antes de comenzar a diseñar la base de datos, es fundamental definir claramente el propósito que servirá. Esto implica comprender qué tipo de datos se almacenarán, cómo se utilizarán y quiénes serán los usuarios principales de la base de datos. Esta fase también puede involucrar la identificación de requisitos específicos, como el tipo de consultas que se ejecutarán frecuentemente.

Paso 2: Identificación de Entidades y Atributos

En el contexto de bases de datos relacionales, una entidad se refiere a un objeto o concepto real sobre el cual queremos almacenar información. Cada entidad se convierte en una tabla dentro de la base de datos. Los atributos son las propiedades o características de las entidades que se convertirán en las columnas de las tablas. Por ejemplo, para una base de datos de gestión de ventas, las entidades podrían ser "Clientes", "Productos" y "Pedidos", con atributos como "Nombre", "Precio" y "Fecha de Pedido". Ademas, en este momento también vamos a identificar aquellos atributos que sean representativos de estas entidades, distinguiendo también cuales podrían ser claves únicas (o claves candidatas).

Paso 3: Creación del Modelo Entidad-Relación

El siguiente paso es el diseño del modelo entidad-relación (ER). Este modelo gráfico muestra cómo se relacionan las distintas entidades en la base de datos. Durante esta fase, se identifican las relaciones entre las entidades, que pueden ser de uno a uno, uno a muchos, o muchos a muchos. Un modelo ER bien diseñado ayuda a visualizar la estructura de la base de datos y a identificar posibles problemas antes de su implementación.

Paso 4: Definición de Claves Primarias y Foráneas

Cada tabla en una base de datos relacional debe tener una clave primaria (Primary Key, PK), que es un identificador único para cada registro. Además, para establecer relaciones entre tablas, se utilizan claves foráneas (Foreign Key, FK), que son campos en una tabla que se refieren a la clave primaria de otra tabla. Las claves foráneas son esenciales para mantener la integridad referencial, asegurando que los datos relacionados se mantengan consistentes y que las referencias cruzadas sean válidas. Y también las UK que son las claves únicas que hemos detectado en nuestro modelo.

Paso 5: Normalización de la Base de Datos

La normalización es el proceso de estructurar una base de datos para minimizar la redundancia de datos y mejorar su integridad. Esto se logra dividiendo la información en tablas relacionadas y eliminando las dependencias anómalas. Las formas normales (1NF, 2NF, 3NF, etc.) son una serie de reglas que guían este proceso. Por ejemplo, en la normalización, se podría dividir una tabla de "Pedidos" que contiene tanto detalles del cliente como del producto en tablas separadas para "Clientes" y "Productos", referenciadas a través de claves foráneas.

Paso 6: Implementación en un Sistema de Gestión de Bases de Datos (SGBD)

Una vez que el diseño está completo, se implementa en un Sistema de Gestión de Bases de Datos (SGBD) como MySQL, PostgreSQL o SQL Server. Este paso implica la creación física de tablas y la definición de las relaciones y restricciones entre ellas. Los comandos SQL son utilizados para crear las tablas, establecer las claves y definir las relaciones entre ellas.

Paso 7: Inserción de Datos y Pruebas

Con la estructura de la base de datos en su lugar, se procede a la inserción de datos iniciales para realizar pruebas. Estas pruebas aseguran que la base de datos funcione como se espera, verificando que las relaciones entre tablas se manejan correctamente y que las consultas devuelven los resultados esperados. Durante esta fase, también se verifica el rendimiento de la base de datos para asegurarse de que puede manejar la carga esperada de usuarios y datos.

Importancia de las Relaciones y la Normalización

Las relaciones y la normalización son fundamentales para el diseño de bases de datos relacionales eficaces. Las relaciones permiten establecer conexiones lógicas entre conjuntos de datos, lo que facilita la realización de consultas complejas y el análisis de datos. La normalización, por su parte, ayuda a mantener la integridad de los datos al eliminar duplicidades y asegurar que cada pieza de información se almacene en un solo lugar. Esto no solo optimiza el uso del espacio de almacenamiento sino que también mejora la consistencia y precisión de los datos.

Consultas y Operaciones

Consultas SQL Complejas

El Lenguaje de Consulta Estructurado (SQL, por sus siglas en inglés) son una serie de comandos para poder interactuar con los Sistemas de Gestión de Base de datos para

realizar las operaciones de gestión de nuestros datos (DML), como de gestión de objetos (DDL), gestión de usuarios y permisos (DCL) y gestión de transacciones (TCL).

Si estamos gestionando datos, utilizamos el lenguaje DDL para poder interactuar con la base de datos. Si bien las consultas simples pueden recuperar datos específicos de una sola tabla, las consultas más complejas permiten extraer datos significativos de múltiples tablas, realizar análisis detallados, y transformar la información en conocimiento útil para la toma de decisiones. A continuación, se presenta una descripción detallada de cómo se pueden realizar consultas SQL más complejas y cómo estas pueden ser utilizadas para extraer datos valiosos de una base de datos.

1. Selección de Datos con Condiciones Avanzadas

Una de las funcionalidades más básicas pero esenciales de SQL es la capacidad de seleccionar datos utilizando condiciones avanzadas para filtrar los resultados. Esto se logra con la cláusula `WHERE`, que puede incorporar operadores lógicos y de comparación.

- **Operadores de Comparación**: Permiten comparar valores en las columnas, como `=`, `<>`, `>`, `<`, `>=`, `<=`.
- **Operadores Lógicos**: `AND`, `OR`, y `NOT` se utilizan para combinar múltiples condiciones, lo que permite crear filtros detallados.
- **Consultas con Rango y Patrones**: Utilizando `BETWEEN` para seleccionar valores dentro de un rango, y `LIKE` para buscar patrones dentro de cadenas de texto.

Ejemplo: Recuperar todos los clientes mayores de 30 años que viven en una ciudad específica.

```
SELECT * FROM Clientes WHERE Edad > 30 AND Ciudad = 'Madrid';
```

2. Uso de Funciones Agregadas

SQL proporciona funciones agregadas que permiten realizar cálculos sobre un conjunto de valores y devolver un único valor. Estas son cruciales para obtener resúmenes de datos, como totales y promedios.

- **SUM()**: Suma todos los valores de una columna.
- **AVG()**: Calcula el promedio de los valores.
- **COUNT()**: Cuenta el número de filas.
- **MAX()** y **MIN()**: Encuentran el valor máximo y mínimo respectivamente.

Ejemplo: Calcular el total de ventas y el promedio de las ventas de un cliente específico.

```
SELECT  SUM(TotalVentas) AS Total, AVG(TotalVentas) AS Promedio FROM Ventas

WHERE ClienteID = 23;
```

3. Agrupación de Datos

La cláusula `GROUP BY` se utiliza para agrupar filas que tienen los mismos valores en columnas especificadas en grupos resumidos, a menudo en combinación con funciones agregadas que van a actuar sobre el grupo que se haya definido. Esto es útil para crear informes que requieren resúmenes de datos.

Ejemplo: Mostrar la cantidad de productos vendidos por categoría. SELECT Categoria, COUNT(*) AS NumeroDeProductos FROM Productos GROUP BY Categoria;

4. Consultas de Unión y Subconsultas

Las consultas más complejas a menudo requieren combinar datos de múltiples tablas. SQL ofrece varias maneras de lograr esto:

- **Unión (JOIN)**: Se utiliza para combinar filas de dos o más tablas, basándose en una columna relacionada entre ellas. Existen varios tipos de uniones:
    - **INNER JOIN**: Devuelve filas cuando hay una coincidencia en ambas tablas.
    - **LEFT JOIN**: Devuelve todas las filas de la tabla izquierda, y las filas coincidentes de la tabla derecha.
    - **RIGHT JOIN**: Devuelve todas las filas de la tabla derecha, y las filas coincidentes de la tabla izquierda.
    - **FULL OUTER JOIN**: Devuelve filas cuando hay una coincidencia en una de las tablas.

Ejemplo: Obtener la lista de clientes y sus pedidos, incluso si no han realizado pedidos. SELECT Clientes.Nombre, Pedidos.Fecha FROM Clientes LEFT JOIN Pedidos ON Clientes.ClienteID = Pedidos.ClienteID;

- **Subconsultas**: Una consulta dentro de otra consulta. Se pueden usar en `SELECT`, `FROM`, `WHERE`, `HAVING`, y `INSERT`.

Ejemplo: Listar productos cuyo precio está por encima del promedio. SELECT NombreProducto FROM Productos WHERE Precio > (SELECT AVG(Precio) FROM Productos);

5. Ordenación y Limitación de Resultados

Para mejorar la presentación de datos, SQL permite ordenar y limitar los resultados.

- **ORDER BY**: Ordena el conjunto de resultados por una o más columnas, de forma ascendente (`ASC`) o descendente (`DESC`).
- **LIMIT:** Limita la cantidad de resultados que se va a devolver al cliente. Junto con el `LIMIT` podemos utilizar el modificador `OFFSET` que nos va a indicar desde que registro queremos obtener los valores (especialmente útil cuando queremos paginar)

Ejemplo: Obtener los 5 productos más caros. SELECT NombreProducto, Precio FROM Productos ORDER BY Precio DESC LIMIT 5;

Ejemplo: Obtener los 5 productos más caros, pero partiendo desde el registro 6. SELECT NombreProducto, Precio FROM Productos ORDER BY Precio DESC LIMIT 5 OFFSET 5;

6. Consultas con Funciones de Ventana

Las funciones de ventana permiten realizar cálculos que utilizan información de varias filas relacionadas en una consulta, sin la necesidad de agrupar los resultados en una única fila.

- **OVER()**: Define una ventana de filas para las funciones de ventana.

Ejemplo: Calcular el salario acumulado para cada empleado, ordenado por fecha de contratación. SELECT EmpleadoID, FechaContratacion, SUM(Salario) OVER (ORDER BY FechaContratacion) AS SalarioAcumulado FROM Empleados;

7. Optimización de Consultas

Para asegurar que las consultas complejas sean eficientes y rápidas, es importante considerar:

- **Índices**: Mejoran la velocidad de las operaciones de búsqueda.
- **Análisis de Plan de Ejecución**: Evaluar cómo SQL ejecuta la consulta para identificar cuellos de botella.
- **Uso de Alias**: Simplifican la consulta y mejoran la legibilidad.

Conclusión

Las consultas SQL complejas son una herramienta poderosa para extraer datos significativos de una base de datos. Al dominar la combinación de operadores, funciones agregadas, uniones, subconsultas y funciones de ventana, se pueden realizar análisis detallados y obtener información valiosa que ayuda a la toma de decisiones estratégicas en cualquier organización. La optimización de consultas garantiza que estas operaciones sean eficientes, especialmente en bases de datos grandes y complejas.

Material Complementario

- SAKILA_DATA_INSERT.sql | Coderhouse