Parte 1: Origen y Primeras Bases de Datos

1. ¿Qué es una base de datos y cuál es su propósito principal?

- Definición y función principal.

Es una colección organizada de datos que se almacenan y gestionan de forma que permiten su fácil acceso, manipulación y actualización.

- Ejemplos históricos de la gestión de información antes de las bases de datos electrónicas.

Las bases de datos son fundamentales en aplicaciones que requieren el manejo de grandes cantidades de información, como sistemas financieros, plataformas de redes sociales, sitios web de comercio electrónico, entre otros.

2. ¿Cómo se almacenaba la información antes del desarrollo de las primeras bases de datos electrónicas?

- Explica los métodos de almacenamiento en papel y tarjetas perforadas.

Almacenamiento en papel: La información se registraba en documentos físicos, como libros, fichas, y formularios. Se utilizaban sistemas de archivo para organizar y recuperar la información.

Tarjetas Perforadas Las tarjetas perforadas eran hojas de papel con agujeros que representaban datos. Cada tarjeta contenía información específica y se leían mediante máquinas que detectaban los agujeros.

- ¿Cuáles eran las principales limitaciones de estos métodos?

Almacenamiento en papel: Buscar información en grandes volúmenes de documentos requería tiempo y esfuerzo, ocupaban mucho espacio, lo que complicaba su almacenamiento y gestión, los documentos podían extraviarse, dañarse o deteriorarse con el tiempo.

Tarjetas Perforadas: Cada tarjeta podía almacenar solo una cantidad reducida de información, la maquinaria necesaria para leer y procesar tarjetas perforadas era costosa y requería mantenimiento, Las tarjetas eran susceptibles a daños físicos, lo que podía resultar en la pérdida de datos.

3. Describe el papel de IBM en el desarrollo de las primeras bases de datos electrónicas. O Menciona el impacto de los sistemas como IMS (Information Management System).

Para mejorar la gestión de datos, se desarrollaron los primeros sistemas de bases de datos jerárquicas. Estos modelos organizaban los datos en una estructura de árbol, donde cada nodo padre podía tener múltiples nodos hijos, pero cada nodo hijo tenía solo un nodo padre. Un ejemplo de este tipo de sistema es el Información Management System (IMS) de IBM, introducido en 1966. Aunque mejoraron el acceso a los datos, las bases de datos jerárquicas tenían limitaciones como la falta de flexibilidad para representar relaciones complejas.

4. ¿Qué es un sistema de bases de datos jerárquico?

- Explica su estructura y cómo se organizan los datos.

Las bases de datos jerárquicas la información está almacenada de forma jerárquica, enlazando los registros en una estructura en forma de árbol invertido, donde encontramos un nodo raíz, nodo padre que pueden tener entre 0 y varios nodos hijos. Las relaciones entre los datos en este esquema jerarquizado se establecen siempre a nivel físico, donde los nodos representan los tipos de registro y los arcos los tipos de interrelaciones jerárquicas entre los mismos.

Una base de datos jerárquica está formada por una colección o «bosque» de árboles separados. Las bases de datos jerárquicas tienen su origen con el comienzo de la programación lógica, aunque no empezaron a usarse más habitualmente hasta 1992. Durante años fueron uno de los modelos de gestión de bases de datos más utilizados, pero con el tiempo y la aparición de otros modelos más ágiles, han ido cayendo en desuso.

- ¿Qué ventajas y desventajas presentaba este modelo?

Entre las ventajas de una base de datos jerárquica encontramos, entre otras las siguientes:

- Las conexiones dentro del árbol son fijas y hace que la navegación por ellas sea rápida.
- Muestra una estructura de la base de datos fácil de ver y comprender.
- Permite predefinir relaciones, simplificando las variaciones futuras.
- Globalizan la información, es decir, cualquier usuario puede acceder a esta información, que se considera un recurso corporativo que no tiene dueños (hablamos dentro del ámbito de una empresa u organización).
- Permite compartir información.
- Permite mantener la integridad la información.
- Mantiene la independencia de datos.

Desventajas

- Pero como decíamos, la base de datos jerárquica también tiene una serie de desventajas, entre las que encontramos:
- Escasa independencia entre los registros (nodos), puesto que, para acceder a un registro, se debe pasar por los padres, algo que quita flexibilidad a la navegación por la base de datos.
- Implica una mala gestión de la redundancia de datos, puesto que cuando un registro tiene relación con dos o más registros, debe almacenarse varias veces, puesto que un hijo no puede tener varios padres.
- Lo anterior implica un mayor volumen de datos y posibles problemas en la integridad y coherencia de los datos, puesto que, si se modifica una de las copias de un registro, se deben modificar también las restantes.
- Sin embargo, modificar este tipo de bases de datos resulta complejo por su rigidez y exige un conocimiento muy amplio sobre la forma en que se han almacenado los datos.

- Diseñar esta base de datos jerárquica requiere conocer muy bien las unidades de información y las relaciones que tienen estas entre sí.

Parte 2: Evolución hacia los Modelos Relacionales

5. ¿Qué innovaciones trajo el modelo relacional propuesto por Edgar F. Codd en 1970?

- Explica en qué consiste este modelo.

En 1970, **Edgar F. Codd**, un científico de IBM, propuso el **modelo relacional** en su artículo "A Relational Model of Data for Large Shared Data Banks". Este modelo revolucionó el manejo de bases de datos al proponer una forma de almacenar datos en tablas bidimensionales (relaciones) donde los datos podían ser manipulados utilizando álgebra relacional.

- ¿Qué beneficios ofrecía en comparación con los modelos jerárquicos o de red?

El modelo relacional introdujo conceptos importantes:

- **Independencia de los datos**: Los usuarios no necesitan conocer el almacenamiento físico de los datos.
- **Lenguaje de consulta estructurado (SQL)**: Introducido por IBM a mediados de los 70, SQL (Structured Query Language) se convirtió en el estándar para interactuar con bases de datos relacionales. La adopción del modelo relacional creció rápidamente con el lanzamiento de sistemas de gestión de bases de datos (DBMS) comerciales como **System R** de IBM y **Ingres** de la Universidad de California en Berkeley.

6. ¿Qué es SQL y por qué fue clave en la adopción del modelo relacional?

- Explica brevemente el lenguaje SQL.

El lenguaje de consulta estructurada (SQL) es un lenguaje de programación para almacenar y procesar información en una base de datos relacional. Una base de datos relacional almacena información en forma de tabla, con filas y columnas que representan diferentes atributos de datos y las diversas relaciones entre los valores de datos. Puede usar las instrucciones SQL para almacenar, actualizar, eliminar, buscar y recuperar información de la base de datos. También puede usar SQL para mantener y optimizar el rendimiento de la base de datos.

- ¿Qué características de SQL lo hicieron fundamental para el manejo de bases de datos?

El lenguaje de consulta estructurada (SQL) es un lenguaje de consulta popular que se usa con frecuencia en todos los tipos de aplicaciones. Los analistas y desarrolladores de datos aprenden y usan SQL porque se integra bien con los diferentes lenguajes de programación. Por ejemplo, pueden incrustar consultas SQL con el lenguaje de programación Java para crear aplicaciones de procesamiento de datos de alto rendimiento con los principales sistemas de bases de datos SQL, como Oracle o MS SQL Server. Además, SQL es muy fácil de aprender, ya que en sus instrucciones se utilizan palabras clave comunes en inglés.

Los sistemas de administración de bases de datos relacionales utilizan un lenguaje de consulta estructurada (SQL) para almacenar y administrar datos. El sistema almacena varias tablas de bases de datos que se relacionan entre sí. MS SQL Server, MySQL o MS Access son ejemplos de sistemas de administración de bases de datos relacionales

7. Compara las bases de datos relacionales con los modelos jerárquicos y de red.

- En términos de flexibilidad, facilidad de uso y rendimiento, ¿qué diferencias encuentras?

Las bases de datos relacionales y no relacionales son dos métodos de almacenamiento de datos para aplicaciones.

Flexibilidad y facilidad de uso: Una base de datos relacional (o base de datos SQL) almacena los datos en formato tabular con filas y columnas. Las columnas contienen atributos de datos, mientras que en las filas hay valores de datos. Se pueden vincular las tablas de una base de datos relacional para obtener información más profunda sobre la interconexión entre diversos puntos de datos. Por otra parte, las bases de datos no relacionales (o bases de datos NoSQL) utilizan diversos modelos de datos para acceder a estos y administrarlos. Están optimizadas específicamente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles, lo que se logra mediante la flexibilización de algunas de las restricciones de coherencia de datos en otras bases de datos.

Las bases de datos relacionales almacenan datos en forma tabular y siguen reglas estrictas sobre las variaciones de datos y las relaciones entre tablas. Le permiten procesar consultas complejas sobre datos estructurados y, al mismo tiempo, mantener la integridad y la coherencia de los datos.

Las bases de datos no relacionales son más flexibles y útiles para datos con requisitos cambiantes. Puede utilizarlas para almacenar imágenes, videos, documentos y otro contenido semiestructurado y no estructurado.

Rendimiento: El rendimiento de las bases de datos relacionales depende de su subsistema de disco. Para mejorar el rendimiento de la base de datos, puede usar SSD y optimizar el disco configurándolo con una matriz redundante de discos independientes (RAID). Para obtener el máximo rendimiento, también debe optimizar los índices, las estructuras de tablas y las consultas.

Por el contrario, el rendimiento de las bases de datos NoSQL depende de la latencia de la red, el tamaño del clúster de hardware y la aplicación que realiza la llamada. Hay algunas maneras de mejorar el rendimiento de una base de datos no relacional:

- Aumentar el tamaño del clúster
- Minimizar la latencia de la red
- Índice y caché

Las bases de datos NoSQL ofrecen un mayor rendimiento y escalabilidad para casos de uso específicos en comparación con una base de datos relacional.