PARTE A

OLTP

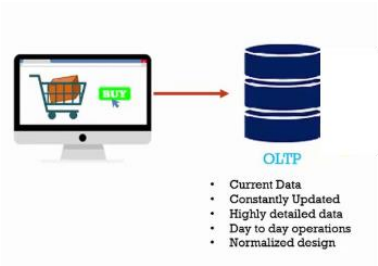
Los sistemas OLTP son bases de datos orientadas al procesamiento de transacciones. Una transacción genera un proceso, y que puede involucrar operaciones de inserción, modificación y borrado de datos.

Análisis OLTP

Los procesamientos de transacciones en línea aportan diversos beneficios algunos de ellos pueden ser:

La simplicidad y la optimización de la tecnología OLTP
Ejemplo de eficiencia y mayor satisfacción para los clientes
Mayor efectividad en la resolución de problemas

Ejemplos:



Banca online
Reservas de tickets de vuelos online
Envió de mensajes
Carritos de compras online

OLAP

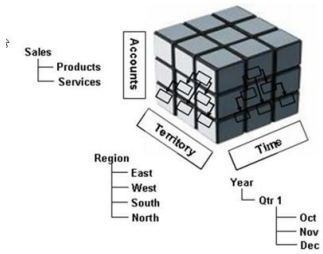
Es una solución utilizada en el campo de la llamada inteligencia empresarial (o Business Intelligence) cuyo objetivo es agilizar la consulta de grandes cantidades de datos. Para ello utiliza estructuras multidimensionales (o cubos OLAP) que contienen datos resumidos de grandes bases de datos o Sistemas Transaccionales (OLTP). Se usa en informes de negocios de ventas, marketing, informes de dirección, minería de datos y áreas similares.

Análisis OLAP

Algunos beneficios pueden ser:

Distribución de la información ágil y Distribución de la información ágil y rápido
Ágil toma de decisiones
Decisiones inteligentes

Ejemplos:



Análisis de ventas en un periodo de tiempo en diferentes regiones
Análisis de compras de los clientes para generar vistas personalizadas acorde a las preferencias

DATAWAREHOUSING

Datawarehouse es un sistema de información que contiene datos históricos y conmutativos de fuentes únicas o múltiples. Simplifica el proceso de informes y análisis de la organización.

Datawarehouse se basa en un servidor RDBMS, que es un depósito de información central rodeado de algunos componentes clave para que todo el entorno sea funcional, manejable y accesible.

Análisis DATAWAREHOUSING

Un Data Warehouse está orientado a asignaturas, ya que ofrece información sobre el tema en lugar de las operaciones en curso de la organización.

En Data Warehouse, integración significa el establecimiento de una unidad de medida común para todos los datos similares de las diferentes bases de datos.

El Data Warehouse tampoco es volátil, lo que significa que los datos anteriores no se borran cuando se ingresan nuevos datos.

Un Datawarehouse es una variante de tiempo ya que los datos en un DW tienen una alta vida útil.

Ejemplos

Realización de minería de datos para obtener nuevos conocimientos de la información contenida en muchas bases de datos grandes

Realizar investigaciones de mercado mediante el análisis en profundidad de grandes volúmenes de datos.

Un negocio en línea que analiza el comportamiento del usuario para tomar decisiones comerciales

DATA LAKES

Un Data Lake es un repositorio de almacenamiento que puede almacenar gran cantidad de datos estructurados, semiestructurados y no estructurados. Es un lugar para almacenar todo tipo de datos en su formato nativo sin límites fijos en el tamaño de la cuenta o el archivo. Ofrece una gran cantidad de datos para aumentar el rendimiento analítico y la integración nativa.

Análisis DATA LAKES

Los Data Lakes tienen las siguientes capacidades:

Capturar y almacenar datos sin procesar a escala por un bajo costo
Almacenar muchos tipos de datos en el mismo repositorio
Realizar transformaciones en los datos
Definir la estructura de los datos en el momento en que se utilizan, denominado esquema en la lectura
Realizar nuevos tipos de procesamiento de datos
Realizar análisis de un solo sujeto basados en Ejemplos muy específicos

Ejemplos:

Data Lakes se pueden utilizan para almacenar:
Datos de flujo de clics
Registros del servidor
Medios de comunicación social
Coordenadas de geolocalización
Datos de máquina y sensor

BIG DATA

Big Data es una colección de grandes conjuntos de datos que no pueden procesarse adecuadamente utilizando técnicas de procesamiento tradicionales. Big data no es solo información, se ha convertido en un tema completo, que involucra varias herramientas, técnicas y marcos.

Big data ayuda a analizar los conceptos en profundidad para las mejores decisiones y estrategias tomadas para el desarrollo de la organización.

Análisis BIG DATA

La importancia de los Big Data es cómo utiliza los datos que posee. Los datos se pueden obtener de cualquier fuente y analizarlos para resolverlos que nos permitan en términos de

Reducciones de costos
Reducciones de tiempo,
Desarrollo de nuevos productos y ofertas optimizadas, y
Toma de decisiones inteligente.

Ejemplos

Bancos
Gobierno
Educación
Salud
Fabrica

FAST DATA

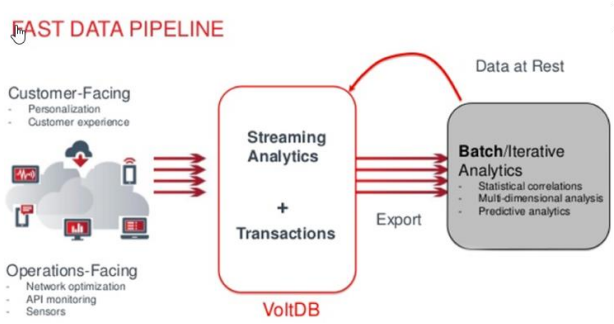
Fast Data son datos en tiempo real que generalmente provienen de la transmisión, como a través de tecnologías de Internet de las cosas (IoT) y aplicaciones basadas en eventos, y se analizan rápidamente para tomar decisiones comerciales rápidas.

Análisis FAST DATA

Fast Data son datos en movimiento, que se transmiten a aplicaciones y entornos informáticos desde cientos de miles hasta millones de puntos finales: dispositivos móviles, redes de sensores, transacciones financieras, feeds de acciones, registros, sistemas minoristas, enrutamiento de llamadas de telecomunicaciones y sistemas de autorización, y más. Las aplicaciones en tiempo real creadas sobre datos rápidos están cambiando el juego para las empresas que dependen de los datos: telecomunicaciones, servicios financieros, salud / medicina, energía y otros. También está cambiando el juego para los desarrolladores, que deben crear aplicaciones para manejar flujos cada vez mayores de datos.

Fast Data al contrario de Big Data, son datos de transmisión: datos en movimiento. Los fast data requieren ser tratados a medida que se transmiten a la empresa en tiempo real. Los Big Data pueden tratarse en otro momento, generalmente después de que se hayan almacenado en un almacén de datos de Hadoop, y analizarse mediante el procesamiento por lotes.

Ejemplos:



Fast Data es útil para procesar datos en tiempo real y recomendaciones, análisis y decisiones (transacciones) en milisegundos (autorización de facturación y aumento de nivel de servicio, por ejemplo, en telecomunicaciones), aunque algunos Ejemplos pueden tolerar hasta minutos de latencia (redes de sensores de energía, por ejemplo).

Ingestión rápida de millones de eventos de datos: transmisiones de datos en vivo desde múltiples puntos finales

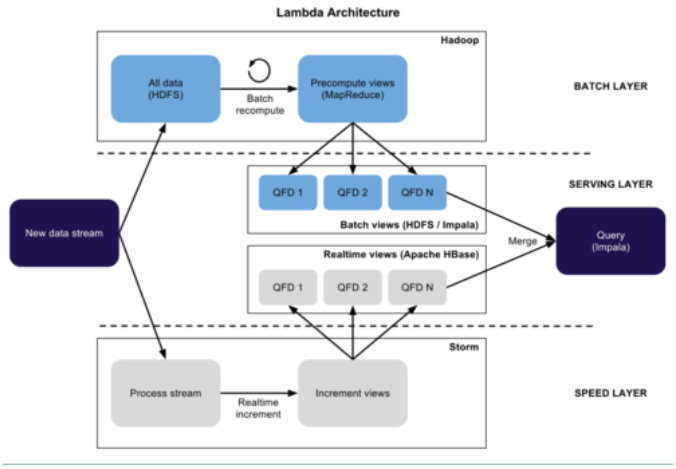
Análisis de transmisión de datos entrantes

Transacciones por evento realizadas en transmisiones en vivo de datos en tiempo real a medida que llegan los eventos.

Arquitectura Lambda

Es una arquitectura de procesamiento de datos genérica, escalable y tolerante a fallas. El objetivo de esta arquitectura es satisfacer las necesidades de un sistema robusto que sea tolerante a fallas, tanto contra fallas de hardware como errores humanos, que pueda atender una amplia gama de cargas de trabajo y Ejemplos, y en el que se requieran lecturas y actualizaciones de baja latencia. El sistema resultante debería ser linealmente escalable, y debería escalar en lugar de aumentar.

Análisis Arquitectura Lambda



Todos los datos que ingresan al sistema se envían tanto a la capa de lote como a la capa de velocidad para su procesamiento.

La capa de lote tiene dos funciones: Administre el conjunto de datos maestro, un conjunto de datos sin procesar inmutable y de solo agregado. Pre-calcular las vistas por lotes.

La capa de publicación indexa las vistas por lotes para que puedan consultarse de forma ad hoc y de baja latencia.

La capa de velocidad compensa la alta latencia de las actualizaciones de la capa de servicio y solo se ocupa de datos recientes, proporcionando vistas en tiempo real.

Cualquier consulta entrante se puede responder combinando resultados de vistas por lotes y vistas en tiempo real.

Ejemplos:

Ejemplo de secuencia de datos:

Smartphones como dispositivos iPhone o Android que tienen una docena de sensores físicos, incluidos GPS y acelerómetro

Wearables como Google Glass que tienen mayor densidad y volumen de sensores

Sensores físicos en edificios (como dispositivos de monitoreo de energía ofrecidos por Nest / Google 4) y en sistemas de transporte (desde automóviles hasta trenes)

Secuencias de redes sociales como las disponibles en la manguera de incendios de Twitter, aplicaciones empresariales como Salesforce o API de agregador, incluido DataSift5

Elementos importantes para optimizar un BD relacional.

El objetivo es hacer aplicaciones que corran más rápido, mejorar (reducir) el tiempo de respuesta de las consultas y transacciones, y mejorar también el ancho de banda de las transacciones.

1. Optimiza los índices:

Tener una buena relación de índices entre tablas es básico para las búsquedas relacionales funcionen correctamente. Agrega índices a las tablas y, sobre ellas, utiliza las sentencias de consulta (SELECT, WHERE…). También resulta recomendable acostumbrarse a verificar periódicamente el registro de consultas (o «queries») lentas para identificar aquellas que deben ser optimizadas.

2. Diseño de la Base de datos:

Desnormalización, que es una desviación de mantener todas las tablas como relaciones BCNF. Si un diseño de base de datos físico no cumple con los objetivos esperados, el DBA puede volver al diseño de la base de datos lógica, hacer ajustes como desnormalizaciones en el esquema lógico y reasignarlo a un nuevo conjunto de tablas e índices físicos.

3. No almacenar imágenes en la base de datos:

Sólo referencias a la ruta en la que se encuentran y metadatos para identificarlas. No nos engañemos, las bases de datos crecen y crecen y las imágenes siguen aumentando en tamaño a su propio ritmo. Si mantienes sólo las referencias al almacén de imágenes, la base de datos estará en forma para devolverse en los procesos más rápidamente.

4. Evite usar comodines (%) al comienzo de un predicado

El predicado LIKE '% abc' causa un escaneo completo de la tabla. Por ejemplo:

```
SELECT * FROM TABLE1 WHERE COL1 LIKE '%ABC'Copy
```

En la mayoría de los casos, este uso de comodines conlleva una importante limitación de rendimiento.

5. Evite columnas innecesarias en la cláusula SELECT

En lugar de usar "SELECT *", siempre especifique columnas en la cláusula SELECT para mejorar el rendimiento de MySQL. Debido a que las columnas innecesarias causan una carga adicional en la base de datos, ralentizan su rendimiento y también todo el proceso sistemático.

6. Use la unión interna, en lugar de la unión externa si es posible

Use la unión externa solo cuando sea necesario. Usarlo innecesariamente no solo limita el rendimiento de la base de datos, sino que también limita las opciones de optimización de consultas de MySQL, lo que resulta en una ejecución más lenta de las declaraciones SQL

7. Dividir las bases de datos en múltiples discos duros

La Entrada / Salida del disco duro se encuentra entre los recursos más lentos de una computadora, lo que se hace evidente a medida que aumenta el tamaño de su base de datos. Muchas bases de datos permiten a los usuarios dividir su base de datos en múltiples discos duros físicos. De hecho, algunos incluso van un paso más allá y permiten dividir el contenido de

una tabla en varios discos. Cuando usa múltiples discos físicos, las operaciones de Entrada / Salida se aceleran significativamente ya que más cabezas obtienen datos en paralelo.

8. Seleccionar datos limitados

Cuantos menos datos se recuperen, más rápido se ejecutará la consulta. En lugar de filtrar en el cliente, inserte la mayor cantidad de filtrado posible en el servidor. Esto dará como resultado que se envíen menos datos por cable y verá resultados mucho más rápido. Eliminar cualquier columna obvia o calculada. Considere el siguiente ejemplo.

9. Descartar índices antes de cargar datos

Considere colocar los índices en una tabla antes de cargar un gran lote de datos. Esto hace que la instrucción de inserción se ejecute más rápido. Una vez que se completan las inserciones, puede volver a crear el índice.

10. Evite usar funciones en predicados

La base de datos no usa índice si tiene alguna función predefinida en la columna.

```
SELECT * FROM TABLE1 WHERE UPPER(COL1)='ABC'Copy
```

Debido a la función UPPER (), la base de datos no utiliza el índice en COL1. Si no hay forma de evitar esa función en SQL, tendrá que crear un nuevo índice basado en funciones o generar columnas personalizadas en la base de datos para mejorar el rendimiento.

11. Use DISTINCT y UNION solo si es necesario

El uso de operadores UNION y DISTINCT sin ningún propósito importante provoca una ordenación no deseada y una ralentización de la ejecución de SQL. En lugar de UNION, el uso de UNION ALL brinda más eficiencia en el proceso y mejora el rendimiento de MySQL con mayor precisión.

12. La cláusula ORDER BY es obligatoria en SQL si espera obtener un resultado ordenado

La palabra clave ORDER BY ordena el conjunto de resultados en columnas de instrucciones predefinidas. Aunque la declaración trae ventaja para los administradores de la base de datos para obtener los datos ordenados, pero también produce un impacto en el rendimiento de bits en la ejecución de SQL. Debido a que la consulta primero necesita ordenar los datos para producir el conjunto de resultados final, provocando una operación un poco compleja en la ejecución de SQL.