Nombres:

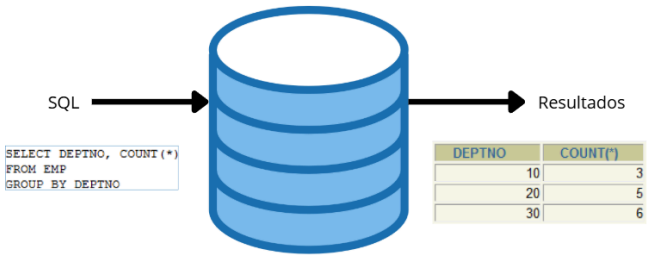
- Laura Isabel Molano Bermúdez
- Nicolás Rocha Gutiérrez
- Luis Eduardo Fierro Ortiz
- Jose David Hurtado Aranda

Tema 5: Procesamiento de Consultas (caso práctico Oracle o SQL Server) incluir, además: Pasos para procesar consultas, b. Medidas del coste de una consulta c. Cómo se procesa una consulta tipo SELECT).

Temas a tratar en cada exposición (los que aplique de acuerdo al tema asignado)
1. Definición.
2. Características.
3. Usos y aplicación.
4. Cómo se monta, qué se instala, cuáles son los requerimientos mínimos. NO APLICA
5. Ejemplos de implementación mínimo 4 ejemplos. No aplica en esta entrega
6. Ventajas y desventajas
7. Tendencias
8. Conclusiones
9. Actividad de evaluación (como los de las pausas activas)

1. Investigación y definición:

Usualmente, cuando se habla de bases de datos, se suele imaginar un cilindro, que es la forma estándar de representarlas. Generalmente, se considera simplemente como un lugar donde se ingresan consultas SQL y se obtiene un resultado. Sin embargo, rara vez se reflexiona sobre lo que ocurre dentro de ese cilindro y cómo se llega a generar ese resultado. Precisamente, eso es lo que se analizará en esta ocasión.



Internamente, en una base de datos existen dos grandes grupos: el motor de ejecución de consultas y el subsistema de almacenamiento. El objetivo del motor de ejecución de consultas es procesar las sentencias SQL, mientras que el subsistema de almacenamiento se encarga de mantener la información de manera coherente en un sistema de ficheros o en memoria, además de prevenir accesos no autorizados a los datos.

El enfoque estará en el motor de ejecución. Cuando un sistema gestor de bases de datos recibe una sentencia SQL, generalmente se siguen tres pasos.

Compilador de LMD:

1. El parseo de la query y la resolución

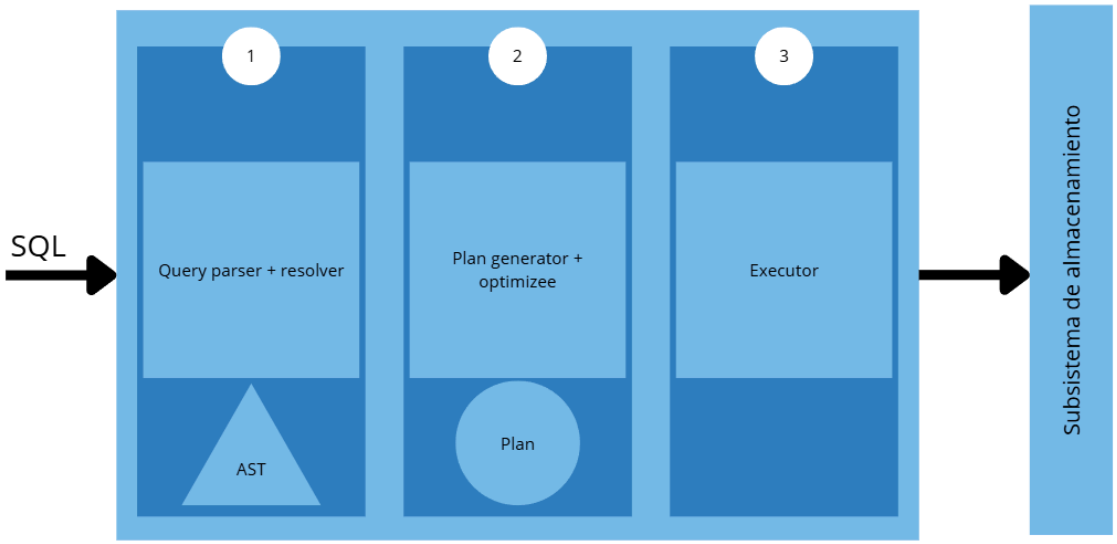
    - **Análisis sintáctico (Parsing)**
    - **Análisis semántico**

2. La generación del plan y sus optimizaciones

    - **Reescritura de la consulta**
    - **Generación del Plan Lógico**
    - **Optimización del Plan Lógico**
    - **Generación del Plan Físico**

Motor de evaluación de consulta:

3. La ejecución



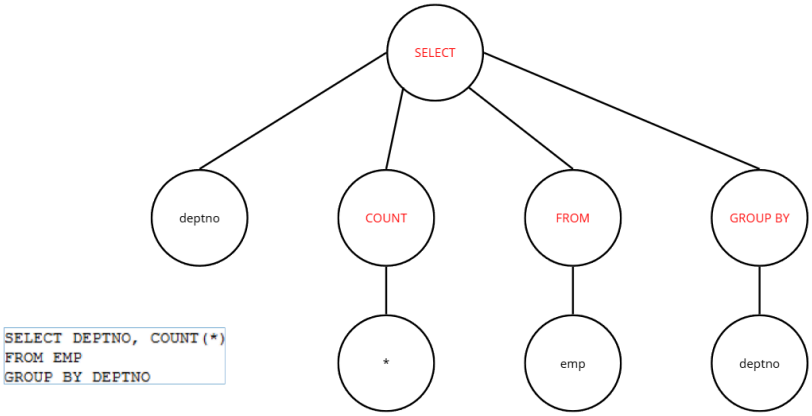
El parseo de la query y la resolución:

- **Análisis Sintáctico (Parsing)**
  Este es el primer paso cuando se analiza una consulta SQL. Su objetivo es comprobar si la consulta está escrita correctamente desde el punto de vista de la gramática del lenguaje SQL y construye el árbol de análisis sintáctico que representa la consulta de forma estructurada con nodos de operadores y nodos de variables.
  Ejemplo sintaxis:

```
SELECT nombre edad
FROM empleados;
```

Detecta que falta una coma entre `nombre` y `edad`.

Ejemplo árbol de análisis sintáctico:



- **Análisis Semántico**
  Después de que la consulta ha pasado el análisis sintáctico, la base de datos verifica su significado.
  - Comprueba que las tablas y columnas mencionadas existen en la base de datos.
  - Verifica que los tipos de datos sean compatibles en las operaciones.
  - Asegurar que los nombres de alias, funciones y operadores sean correctos.

  Ejemplo:

```
SELECT nombre, salario
FROM empleados
WHERE edad = 'treinta';
```

  Detecta que edad es un campo numérico, pero se está comparando con una cadena ('treinta').

La generación del plan y sus optimizaciones

Pasamos de un árbol de sintaxis abstracta a un código ejecutable por el sistema gestor de base de datos para resolver el árbol de sintaxis abstracta, este apartado es el enlace entre la definición y la ejecución. Este apartado vendría dependiendo del sistema gestor que estemos usando pero en general tenemos estos componentes

- **Reescritura de la consulta/query**
  (https://www.ibm.com/docs/es/db2/11.1?topic=process-query-rewriting-methods-examples):

Es una técnica utilizada para transformar una consulta SQL en otra equivalente que sea más eficiente o que se adapte a ciertas restricciones del sistema para mejorar el rendimiento y encontrar el mejor plan de acceso a los datos.
Existen tres formas principales de reescribir consultas:

1. Fusión de operaciones: Se combinan operaciones para evitar cálculos innecesarios.

   Ejemplo: Queremos obtener el número de empleados por departamento y, al mismo tiempo, filtrar aquellos departamentos que tienen empleados con salario superior a 3000.
   Consulta:

```
SELECT deptno, COUNT(*) AS total_empleados
FROM emp
WHERE deptno IN (SELECT DISTINCT deptno FROM emp WHERE sal > 3000)
GROUP BY deptno;
```

   Consulta optimizada:

```
SELECT deptno, COUNT(*) AS total_empleados
FROM emp
WHERE sal > 3000
GROUP BY deptno;
```

   En lugar de hacer una subconsulta para filtrar los departamentos y luego contar los empleados, **fusionamos ambas operaciones en una sola consulta** que cuenta directamente los empleados con salario superior a 3000.

2. Movimiento de operación: Se reorganizan las operaciones para que la consulta se ejecute de manera más eficiente.

   Ejemplo:Queremos calcular el promedio de salario de los empleados en departamentos donde hay al menos un gerente (job = 'MANAGER').
   Consulta:

```
SELECT deptno, TO_CHAR(AVG(sal), '$999,999,999.99') AS promedio_salario
FROM emp
WHERE deptno IN (SELECT DISTINCT deptno FROM emp WHERE job = 'MANAGER')
GROUP BY deptno;
```

   Consulta optimizada:

```
SELECT deptno, TO_CHAR(AVG(sal), '$999,999,999.99') AS promedio_salario
FROM emp
GROUP BY deptno
HAVING SUM(CASE WHEN job = 'MANAGER' THEN 1 ELSE 0 END) > 0;
```

   En lugar de hacer una subconsulta, movemos la condición al HAVING y usamos SUM(CASE WHEN ...) para contar los gerentes dentro de cada departamento. Esto permite que la base de datos haga un solo escaneo sobre emp en lugar de dos.

3. Conversión de predicados: Se transforman ciertas condiciones para optimizar la búsqueda de datos.

   Ejemplo: Queremos obtener los empleados contratados después del 1 de enero de 2020, pero en lugar de comparar directamente la fecha, convertimos hiredate a una cadena.

   Consulta:

```
SELECT empno, ename, hiredate
FROM emp
WHERE TO_CHAR(hiredate, 'YYYY-MM-DD') > '2020-01-01';
```

   Consulta optimizada:

```
SELECT empno, ename, hiredate
FROM emp
WHERE hiredate > DATE '2020-01-01';
```

   En lugar de transformar hiredate en un string con TO_CHAR, la comparación se hace directamente con un DATE, esto permite que Oracle utilice índices en hiredate, mejorando el rendimiento.

incluir ejemplos de optimización con algebra y consultas más elaboradas

- **Generación de un Plan lógico**:
  La consulta generada se traduce al algebra relacional con la finalidad de encontrar la mejor estrategia para lograr una consulta más eficiente

  Selección (σ) ⟶ Filtrar filas (WHERE).
  Proyección (π) ⟶ Seleccionar columnas (SELECT).
  Unión (⋈) ⟶ Relacionar tablas (JOIN).
  Ordenamiento (τ) ⟶ Ordenar filas (ORDER BY).
  Agrupación (γ) ⟶ Agrupar (GROUP BY).

- **Optimización del Plan lógico:**
  En la optimización se organizan las operaciones para minimizar el costo computacional, se aplican reglas como:
  - Aplicar filtros lo antes posible para reducir el número de filas. (Regla de AR)
  - Reordenar JOIN para hacerlos más eficientes. (Regla de AR)
  - Usar índices en lugar de hacer un escaneo completo de tablas. (Análisis necesario para la generación del plan físico)

  Ejemplo:
  Consulta:

```sql
SELECT nombre
FROM empleados
WHERE edad > 30
ORDER BY salario DESC;
```

Plan Lógico Inicial (Álgebra Relacional):

- **Selección (σ)**: σ(edad > 30) (empleados)
- **Proyección (π)**: π(nombre, salario)
- **Ordenamiento (τ)**: τ(salario DESC)

Plan Lógico Optimizado:

- Primero se hace la selección para reducir el número de filas (edad > 30).
- Luego se hace la proyección para reducir las columnas (nombre, salario).
- Finalmente, se hace el ordenamiento (ORDER BY salario DESC).

- **Generación del plan físico**:
  El plan físico se construye en base al plan lógico, pero con decisiones concretas sobre cómo acceder a los datos. Se deben responder preguntas como:
  - ¿Se usará un índice o se hará un escaneo completo de la tabla?
  - ¿Qué tipo de algoritmo de JOIN es más eficiente en este caso?
  - ¿Cómo se manejará la ordenación y agrupación?

  Para responder esto, el optimizador evalúa distintas estrategias y elige la de menor costo según una función matemática llamada costo estimado.

  1. **Seleccionar métodos de acceso a los datos** (https://www.ibm.com/docs/es/db2/11.1?topic=optimization-data-access-methods):
     Aquí se decide cómo se obtendrán los registros. Existen varios métodos:
     - Escaneo secuencial (Full Table Scan)
       Se lee toda la tabla registro por registro. Es ineficiente para grandes volúmenes de datos, pero útil si no hay índices.
     - Búsqueda con índice (Index Scan o Index Seek)
       Se usa un índice para encontrar rápidamente las filas.
       Ejemplo: si buscas WHERE id = 10 y id tiene un índice, solo se leen las filas necesarias.
     - Acceso a través de un índice con búsqueda de tabla (Index Lookup + Table Access):

Primero se usa un índice para encontrar la fila y luego se consulta la tabla completa si hay más columnas necesarias.

Ejemplo:

```
SELECT * FROM empleados WHERE id = 5;
```

Si **id** es una Primary Key con índice, el motor usará el índice para encontrar la fila más rápido.

2. **Elegir el algoritmo de JOIN adecuado:**
   Si la consulta involucra un JOIN, el motor decide qué algoritmo usar. Los principales son:
   - Nested Loop Join (bueno para tablas pequeñas):
     Itera sobre cada fila de una tabla y la compara con las filas de la otra. Útil si hay índices en la tabla más grande.
   - Hash Join (bueno para conjuntos grandes sin índices):
     Se usa una estructura hash para acelerar la comparación de filas. Útil si no hay índices en las columnas de unión.
   - Merge Join (bueno para datos ordenados):
     Funciona eficientemente si ambas tablas ya están ordenadas por la clave de unión.

   Ejemplo:

```
SELECT emp.ename, dept.dname
FROM emp INNER JOIN  dept
ON emp.deptno = dept.deptno;
```

   Si deptno tiene un índice en ambas tablas, es probable que el motor use Nested Loop Join o Merge Join. Si no hay índices, podría hacer un Hash Join.

3. **Optimizar las operaciones de ordenamiento y agrupación**
   Para ORDER BY, GROUP BY, y DISTINCT, el motor elige entre:
   - Ordenamiento con índice (Index Sorting)
     Si la consulta ya usa un índice ordenado, el ordenamiento es rápido.
   - Ordenamiento con Sort Merge
     Si los datos no están ordenados, se usa un algoritmo de ordenación como Merge Sort.
   - Agrupación con Hash Aggregation
     En GROUP BY, si la tabla es grande, puede usarse una estructura hash para agrupar más rápido.
     Ejemplo:

```
SELECT empno, ename, sal
FROM emp
ORDER BY sal;
```

Si sal no tiene un índice, Oracle necesita ordenar todos los registros manualmente con Sort Merge Sort, lo que puede ser costoso si hay muchos registros.

4. **El motor de la base de datos simula diferentes planes físicos** y asigna a cada uno un costo estimado, basado en factores como:
   - Número de registros a leer
   - Uso de índices
   - Tiempo estimado de ejecución

Finalmente, se elige el plan de menor costo para ejecutar la consulta.

EJEMPLO: suponemos que emp tiene edad

```
SELECT ename
FROM emp
WHERE edad > 30
ORDER BY sal DESC;
```

Posibles estrategias del plan físico:

1. Estrategia con escaneo de tabla completa
   - Se lee toda la tabla (Full Table Scan).
   - Se filtra (edad > 30).
   - Se ordena (ORDER BY salario DESC).
2. Estrategia con índice (más rápida)
   - Se usa un índice en edad para filtrar rápido (Index Scan).
   - Se usa un índice en salario para ordenar (Index Sorting).

El optimizador elegirá la segunda estrategia porque tiene menor costo.

Ejecución:

El sistema necesita asignar memoria y procesadores para la consulta. Esto se hace al abrir un cursor, que es un objeto que permite leer los resultados fila por fila si es necesario.

Se reserva memoria para almacenar datos temporales y se asigna el número de hilos de ejecución (puede usar paralelismo en consultas grandes), finalmente se inicia la ejecución del plan físico.

Medidas del Coste de una Consulta en Oracle

El coste de una consulta se refiere a la estimación de los recursos que se necesitarán para ejecutarla, y no se mide en tiempo real. Oracle usa estas medidas de coste para seleccionar el plan de ejecución más eficiente:

1. **Operaciones de Entrada/Salida (E/S):** Estima el número de lecturas y escrituras necesarias. Las operaciones de E/S son una de las formas más lentas de acceder a los datos, por lo que reducir estas operaciones es esencial para mejorar el rendimiento de las consultas.
2. **Uso de CPU**: Representa el tiempo estimado que la consulta utilizará de la CPU. Las consultas con operaciones complejas (como agregaciones o cálculos) tienden a tener un mayor coste en CPU.
3. **Consumo de Memoria:** Estima la cantidad de memoria que se necesita para realizar la consulta. Consultas que manejan grandes volúmenes de datos o necesitan trabajar con múltiples tablas pueden requerir más memoria.

Ejemplo: Un INDEX RANGE SCAN normalmente tiene un coste menor que un FULL TABLE SCAN, ya que accede solo a los datos relevantes utilizando un índice, en lugar de leer toda la tabla.

2. Las características a destacar son :

- Incluye análisis, optimización y ejecución.
- Utiliza planes de ejecución para mejorar el rendimiento.
- Puede involucrar índices, estadísticas y caché.
- Se evalúa el costo computacional de la consulta.

3. Casos de Uso y Aplicaciones del Procesamiento de Consultas

El procesamiento de consultas tiene aplicaciones en varios contextos dentro de la administración de bases de datos Oracle, ayudando a garantizar que las consultas sean eficientes y los recursos se gestionen de manera óptima.

1. **Optimización del Rendimiento**
    - **Caso de uso:** Identificación de consultas lentas que consumen recursos excesivos como CPU y E/S.
    - **Aplicación:** El optimizador evalúa diferentes planes de ejecución y selecciona el más eficiente, minimizando los costes de CPU y E/S, lo que resulta en una mejora del rendimiento.
2. **Manejo de Grandes Volúmenes de Datos**
    - **Caso de uso:** Consultas en bases de datos que contienen millones de filas.
    - **Aplicación:** El uso de particionamiento de tablas y ejecución paralela ayuda a manejar eficientemente grandes volúmenes de datos, reduciendo el tiempo de ejecución de las consultas.
3. **Control y Gestión de Costes de Recursos**

- **Caso de uso:** En sistemas con múltiples usuarios, donde es importante distribuir los recursos de manera equitativa.
- **Aplicación:** Se pueden implementar estrategias de priorización de consultas y limitación de recursos por sesión para garantizar que los usuarios no afecten el rendimiento global del sistema.

4. **Diagnóstico y Solución de Problemas**
   - **Caso de uso: Consultas de diagnóstico** para identificar problemas de rendimiento, especialmente en sistemas con **consultas lentas**.

   - **Aplicación:** El uso de herramientas como `EXPLAIN PLAN` permite ver el **plan de ejecución detallado** y descubrir qué operaciones están consumiendo más recursos. El coste relacionado con el uso de **índices** es fundamental aquí. Si una consulta realiza un `FULL TABLE SCAN`, generalmente tendrá un coste de E/S más alto, ya que está leyendo toda la tabla. En cambio, si se usa un **índice**, el coste puede ser considerablemente menor, pues se accede solo a los datos relevantes sin necesidad de leer toda la tabla.

   - **Coste relacionado con índices:** Cuando Oracle utiliza **índices**, el coste de la consulta disminuye al realizar un acceso más directo a las filas que cumplen con las condiciones de la consulta. Si no hay índices disponibles o adecuados, la base de datos realiza un **FULL TABLE SCAN**, lo que incrementa significativamente el coste de E/S y el tiempo de ejecución. El coste asociado a los **índices** se refiere a la **lectura más eficiente** de los datos, y se calcula evaluando las **hojas del índice** y la **carga de trabajo de las estructuras de árbol** que componen el índice.
5. **Análisis de Consultas Ad-Hoc y Reportes**
   - **Caso de uso:** Consultas que son ejecutadas de manera ad-hoc, sin un patrón predefinido.
   - **Aplicación:** La caché de resultados y el uso de vistas materializadas pueden acelerar las consultas ad-hoc, especialmente en entornos de BI donde se generan informes complejos a partir de datos grandes.
6. **Integración con Otras Herramientas y Servicios**
   - **Caso de uso:** Consultas ejecutadas dentro de plataformas de Business Intelligence (BI).
   - **Aplicación:** Oracle se integra con herramientas como Oracle BI o plataformas de terceros para realizar consultas complejas y generar informes visuales.
7. **Tareas de Mantenimiento y Actualización de Base de Datos**
   - **Caso de uso:** El mantenimiento regular de bases de datos para garantizar que las estadísticas y los índices estén optimizados.
   - **Aplicación**: Se realizan tareas periódicas como la reorganización de índices y la actualización de estadísticas para que el optimizador tenga información precisa al generar planes de ejecución.
8. **Adaptación Automática a Cambios en la Carga de Trabajo**
   - **Caso de uso:** En sistemas de bases de datos con cambios dinámicos en la carga de trabajo.

- **Aplicación:** Oracle ajusta automáticamente los planes de ejecución y la distribución de recursos según el comportamiento de las consultas y la carga de trabajo del sistema.

6. Ventajas Y Desventajas

VENTAJAS	DESVENTAJAS
INTEGRACIÓN: El procesamiento de consultas en SQL Server se integran perfectamente con otros productos de Microsoft como Azure y Visual Studio, facilitando la creación de soluciones de software interconectadas.	**COSTES:** Para un rendimiento óptimo, especialmente con grandes volúmenes de datos, se necesita una infraestructura de hardware robusta, lo que puede incrementar los gastos en infraestructura.
ESCABILIDAD Y RENDIMIENTO: Ambos sistemas (SQL Server y Oracle) pueden manejar grandes volúmenes de datos y transacciones simultáneas sin comprometer el rendimiento, lo que los hace adecuados para empresas de diversos tamaños.	**USO DE RECURSOS:** Consultas complejas pueden requerir grandes cantidades de memoria y procesamiento, afectando el rendimiento del servidor, también pueden generar una alta carga de lectura y escritura en discos duros, ralentizando el sistema si no hay una optimización adecuada.
OPTIMIZACIÓN: Oracle admite la optimización de consultas en estrella, mejorando el rendimiento en consultas complejas de bases de datos multidimensionales.	**RENDIMIENTO:** En situaciones donde se manejan vastas cantidades de datos o grandes volúmenes de consultas simultáneas, las bases de datos SQL pueden enfrentar cuellos de botella de rendimiento, especialmente si no están optimizadas adecuadamente.
LECTURA Y ENTENDIMIENTO: Las consultas, especialmente las SELECT bien estructuradas, suelen ser fáciles de leer y comprender, tanto SQL Server Management Studio Como Oracle SQL Developer permiten visualizar los planes de ejecución, lo que facilita comprender cómo se ejecuta una consulta.	**CURVA DE APRENDIZAJE:** Aunque SQL es estándar, Oracle y SQL Server tienen diferencias en sintaxis y características avanzadas como PL/SQL (Oracle) o T-SQL (SQL Server).

7. Tendencias

Las actuales tendencias en procesamiento de consultas se basan más en el resultado de este procesamiento aunque hay una que resalta mucho tanto en el procesamiento de la consulta como su resultado hablamos de el uso de la ia.

La ia actual permite la optimización de consultas y un procesamiento de datos en tiempo real a día de hoy.

En los próximos años, el mundo del análisis de datos y la inteligencia artificial seguirá cambiando rápidamente, y el procesamiento de datos será un factor clave en esta transformación. Hay dos tendencias principales que están ganando impulso:

- Modelos de Lenguaje Grande (LLMs) y Bases de Datos Vectoriales: Modelos como GPT-4 están revolucionando la forma en que las empresas manejan grandes volúmenes de datos. Estos modelos requieren nuevas formas de almacenar y organizar la información, como bases de datos vectoriales, que son más eficientes para manejar datos complejos. Gracias a estas innovaciones, las empresas podrán analizar datos más rápido y obtener información más precisa, lo que les ayudará a tomar mejores decisiones esta tecnología facilita el manejo de información más compleja y mejora la capacidad de extraer valor de los datos, optimizando los procesos de análisis.
- Analítica en tiempo real: La capacidad de procesar y analizar datos en tiempo real se ha convertido en una necesidad fundamental para las organizaciones. Esta tendencia permite a las empresas responder rápidamente a los cambios en el mercado y anticiparse a nuevas oportunidades o amenazas. La analítica en tiempo real facilita la toma de decisiones basada en datos instantáneos, lo que se vuelve esencial en entornos empresariales dinámicos. Juntas, estas tendencias están dando forma al futuro del procesamiento de datos, donde las empresas tendrán acceso a herramientas más avanzadas para administrar y analizar la información de manera más eficiente y en tiempo real.

https://ai2sql.io/es-es/ia-para-consultas-sql-precisas

https://hub.laboratoria.la/5-tendencias-analisis-de-datos-e-ia-para-el-2025?utm_source=chatgpt.com

8. Conclusiones

- La mayoría de usuarios al utilizar o trabajar en una base de datos no se percatan de la complejidad que hay por debajo de una consulta. Se trabaja automáticamente pensando que una base de datos simplemente es un sistema donde se ingresa una consulta y se obtiene un resultado, pero como se mencionó anteriormente, es algo más complejo donde existe un motor de ejecución que sigue múltiples etapas para transformar la consulta en un plan de ejecución eficiente.
- Las etapas del motor de ejecución son importantes a la hora del procesamiento de las consultas, ya que ayudan a asegurar que la consulta SQL esté correctamente estructurada y tenga un sentido dentro del contexto de la gramática del lenguaje, radicando en la optimización de la consulta, eligiendo diferentes estrategias para lograr lo anterior dicho y también teniendo en cuenta la reducción de operaciones innecesarias, todo esto centrándose claramente en la optimización de los recursos disponibles y el tiempo de respuesta de las consultas.

- Las medidas de coste de una consulta en Oracle (operaciones de entrada y de salida, uso de CPU y consumo de memoria) permiten estimar el impacto que tendrán estas al momento de su ejecución, para minimizar el tiempo de respuesta y el uso de recursos.
- Las aplicaciones del procesamiento de consultas abarcan desde la optimización del rendimiento, la gestión de grandes volúmenes de datos y la distribución eficiente de recursos, hasta la solución de problemas y el mantenimiento del sistema. Haciendo que el procesamiento de consultas sea una tecnología clave para diferentes entornos como los empresariales, educativos y sistemas con grandes números de usuarios.
- En cuanto a las tendencias, el procesamiento de consultas se ha adaptado a tecnologías, como lo son la inteligencia artificial, ahora con el uso de la IA se permite una mayor optimización y procesamiento de datos en tiempo real, se habla de dos tendencias claves las cuales son: Modelos de Lenguaje Grande (LLMs) y Bases de Datos Vectoriales, y Analítica en tiempo real. La primera mejora la capacidad de analizar y gestionar grandes volúmenes de datos, mientras que la segunda ayuda a empresas a tomar decisiones rápidas y precisas.

9. Actividad de evaluación: Ejercicio de preguntas de respuesta múltiple con Quizizz