UNIVERSIDAD ICESI

TU FUTURO A OTRO NIVEL

Sistemas Intensivos en datos 2

Unidad 2- Administración y optimización de DB

Unidad 2 - Administración y Optimización de RDBs

- Parte 1: Atributos de calidad / características de un SGBDR
- Parte 2: Seguridad
- Parte 3:Rendimiento

Parte 1 - Atributos de Calidad

Determinan la utilidad y la eficacia de la base de datos:

- Exactitud: . Los datos precisos son esenciales para tomar decisiones informadas.
- Completitud: Los datos completos son necesarios para tener una imagen completa de la situación.
- Actualidad: Los datos actualizados garantizan que las decisiones estén basadas en la información más reciente.
- Consistencia: Los datos consistentes evitan errores y malentendidos.
- Integridad: Los datos íntegros aseguran que los datos cumplan con los requisitos de la empresa.
- Usabilidad: Los datos fáciles de usar hacen que la base de datos sea más accesible para los usuarios.
- Eficiencia: Los datos eficientes mejoran el rendimiento de la base de datos y reducen los costos.
- Seguridad: Los datos seguros protegen la información de los usuarios y evitan el acceso no autorizado.

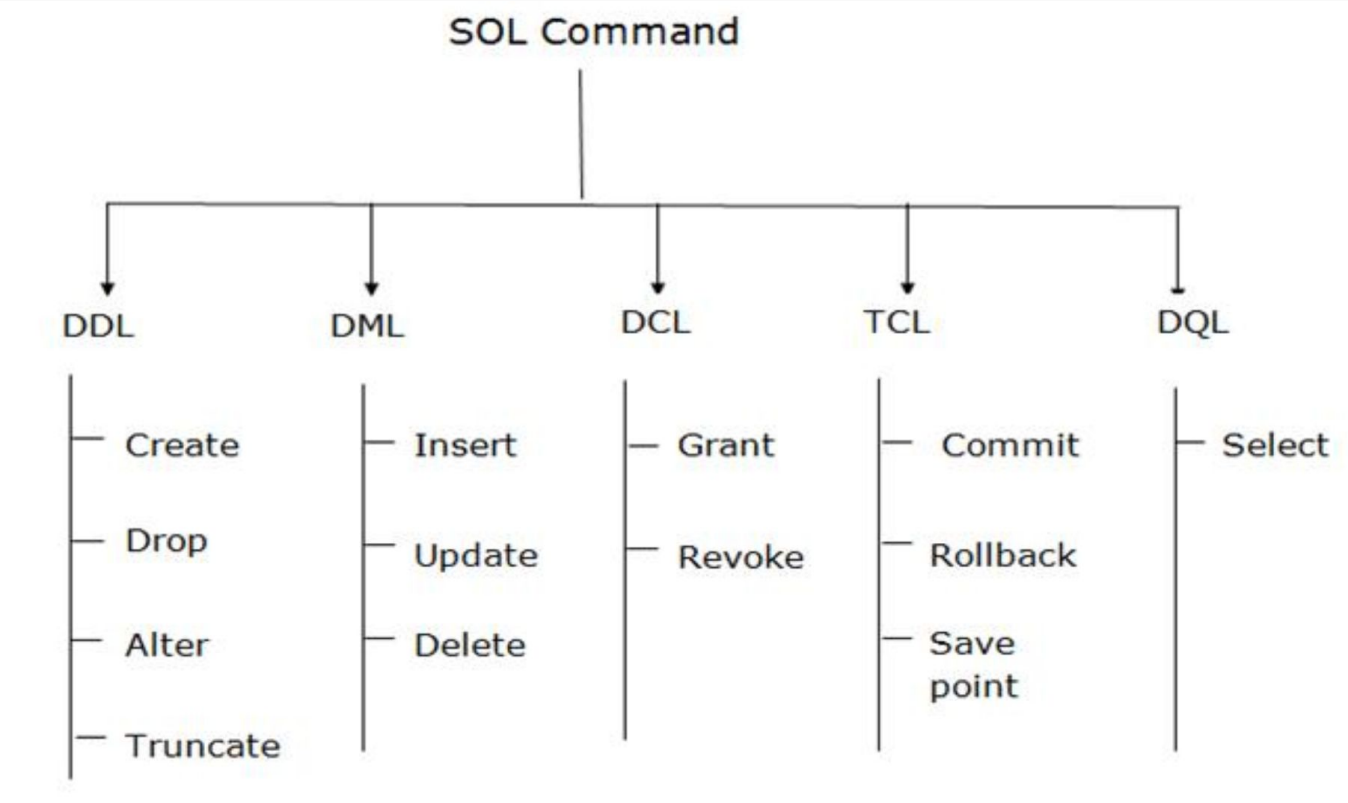
Parte 1 - Atributos de Calidad

- Disponibilidad: Los datos disponibles permiten tomar decisiones en el momento adecuado. 7x24
- Universalidad: Los datos pueden pertenecer a diferentes dominios. Múltiples tipos de datos. Multimedia
- Portabilidad: Los datos y las funcionalidades sobre ellos deben permanecer aún cuando se cambie de versión del SGBD
- Interoperabilidad: Los SGBD pueden operar sobre diferentes o múltiples plataformas / Sistemas operativos
- Escalabilidad: La gestión de los datos debe permitir el incremento de los recursos invertidos.
- Escalabilidad flexible: Incremento o decremento de los recursos invertidos.

Parte 1 - Atributos de Calidad

¿Cómo se implementan los atributos de calidad?

Parte 1 - Atributos de Calidad



Parte 2 - Atributo de seguridad

- Mediante datos encriptados
- Mediante roles y permisos de acceso a los objetos
- Mediante Objetos de la Base de Datos
- Mediante gestión de Transacciones

- Esquemas
- Roles
- Usuarios
- Permisos

- Vistas
- Sinónimos

Esquemas, usuarios, permisos, roles

- Esquema ←→ Usuario (oracle, posgreSQL?)
- Usuarios
- Permisos y roles (otorgar o revocar)

Vistas (Views)

- Puede considerarse una tabla virtual
- Puedo seleccionar algunas columnas de una tabla (Nivel 1 de seguridad)
- Puedo seleccionar algunas filas de una tabla (Nivel 2 de seguridad)
- Puedo unir Nivel 1 y 2 sobre una tabla o sobre múltiples tablas.

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM Student_Details
WHERE STU_ID < 4;
```

Vistas (Views)

- Puedo unir Nivel 1 y 2 sobre una tabla o sobre múltiples tablas.

```
CREATE VIEW MarksView AS
SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS
FROM Student_Detail, Student_Mark
WHERE Student_Detail.NAME = Student_Marks.NAME;

SELECT * FROM MarksView;
```

Sinónimos (Synonyms)

```
CREATE [OR REPLACE] [PUBLIC] SYNONYM schema.synonym_name
FOR schema.object;
```

```
CREATE SYNONYM stocks
FOR inventories;
```

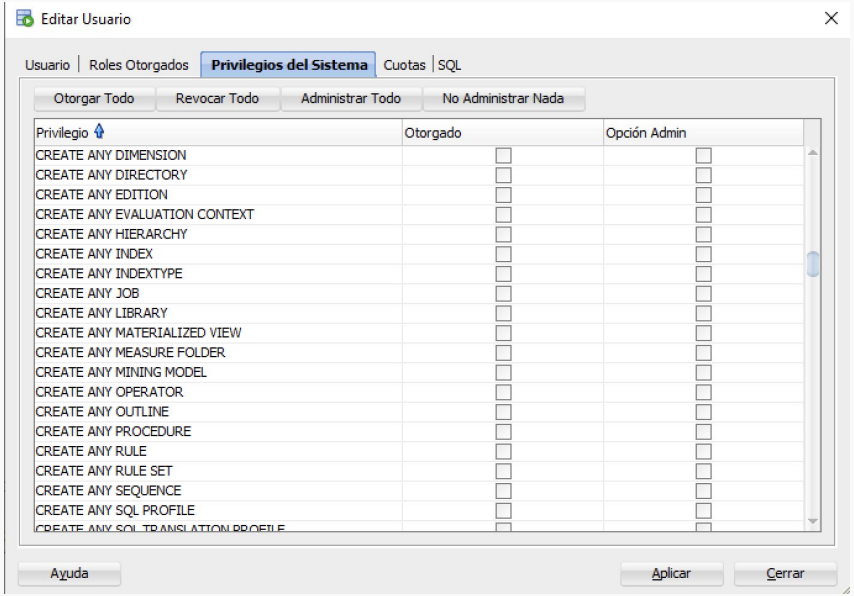
Privilegios y roles

Privilegio: determinan lo que alguien está autorizado a hacer con los datos y la base de datos

Rol: Agrupación de Privilegios

GRANT

REVOKE



Privilegios y roles

GRANT privilege_list ON object_name TO user_name [WITH GRANT OPTION];

GRANT SELECT ON customers TO john;

REVOKE privilege_list ON object_name FROM user_name [CASCADE];

REVOKE SELECT ON customers FROM john;

Parte 3 - Atributo de rendimiento

- Afinamiento (SQL tuning)
- Plan de Ejecución
- Optimización de consultas
- Índices

Afinamiento de SQL (SQL Tuning)

Abarca mucho más allá de afinar las sentencias SQL, cómo identificar declaraciones SQL de alto recurso, qué se debe recopilar para luego ajustar y proporcionar sugerencias de ajuste.

1. Introducción al afinamiento de SQL
2. Objetivos del afinamiento
3. Identificación de SQL de alta carga
4. ~~Funciones de afinamiento/ajuste automático de SQL~~
5. Desarrollo de sentencias SQL eficientes

Introducción al afinamiento de SQL

El ajuste de SQL implica tres pasos básicos:

1. Identificar **declaraciones SQL de carga alta** o principales que son responsables de una gran parte de la carga de trabajo de la aplicación y los recursos del sistema, mediante la **revisión del historial de ejecución de SQL** anterior disponible en el sistema.

2. Verificar que los **planes de ejecución** producidos por el optimizador de consultas para estas declaraciones funcionen de manera razonable.

3. Implementar **acciones correctivas** para generar mejores planes de ejecución para declaraciones SQL de bajo rendimiento.

Estos tres pasos se repiten hasta que el rendimiento del sistema alcanza un nivel satisfactorio o no se pueden ajustar más declaraciones.

Objetivos del afinamiento SQL

El ajuste de SQL involucra tres pasos básicos:

Reducir la carga de trabajo	Balancear la carga de trabajo	Paralelizar la carga de trabajo
Implica encontrar formas más eficientes de procesar la misma carga de trabajo. Ej. uso de índices.	Repartir la carga de trabajo, durante todo el día. Programar informes.	Esto es posible en consultas OLAP, en sistemas OLTP puede perjudicar los tiempos de respuesta. Ej. Informe de un año, se puede partir en trimestres.

Objetivos del afinamiento de SQL

El ajuste de SQL implica tres pasos básicos:

1. Reducir la carga de trabajo

2. Balancear la carga de trabajo

3. Paralelizar la carga de trabajo

Reducir la carga de trabajo:
implica encontrar formas más eficientes de procesar la misma carga de trabajo.

Si una consulta ejecutada comúnmente necesita acceder a un pequeño porcentaje de datos en la tabla, entonces se puede ejecutar de manera más eficiente mediante el uso de un índice. Al crear un **índice** de este tipo, se reduce la cantidad de recursos utilizados.

Si un usuario está mirando las primeras veinte filas de las 10.000 filas devueltas en un orden de clasificación específico, y si la consulta (y el orden de clasificación) pueden satisfacerse mediante un índice, entonces el usuario no necesita acceder y ordenar las 10.000 filas. para ver las primeras 20 filas.

Objetivos del afinamiento de SQL

Balancear la carga de trabajo:

Los sistemas a menudo tienden a tener un uso alto durante el día, cuando los usuarios reales están conectados al sistema, y un uso bajo durante la noche. Si los informes no críticos y los trabajos por lotes se pueden programar para que se ejecuten durante la noche y se puede reducir su simultaneidad durante el día, se liberan recursos para los programas más críticos durante el día.

Paralelizar la carga de trabajo:

Las consultas que acceden a grandes cantidades de datos (consultas típicas de almacén de datos) a menudo se pueden paralelizar. Esto es extremadamente útil para reducir el tiempo de respuesta en almacenes de datos de baja concurrencia. Sin embargo, para los entornos OLTP, que tienden a ser de alta concurrencia, esto puede afectar negativamente a otros usuarios al aumentar el uso general de recursos del programa.

Identificar Altas Cargas de Trabajo

SQL de alta carga son sentencias SQL de bajo rendimiento que consumen muchos recursos y que afectan el rendimiento de la base de datos Oracle. Las sentencias SQL de alta carga se pueden identificar mediante:

- Monitor de diagnóstico automático de bases de datos
- Repositorio automático de cargas de trabajo
- Vista V$SQL
- Carga de trabajo personalizada
- Seguimiento SQL

Afinamiento de aplicaciones

Si toda su aplicación no funciona de manera óptima, o si está intentando reducir la CPU general o la carga de E/S en el servidor de la base de datos, identificar SQL que consume muchos recursos implica los siguientes pasos:

1. Determine qué período del día le gustaría examinar; normalmente este es el tiempo máximo de procesamiento de la aplicación.
2. Recopile estadísticas del sistema operativo y de Oracle al principio y al final de ese período.

Las estadísticas mínimas son:

Paso 1: E/S de archivos (V$FILESTAT),

Paso 2: Estadísticas del sistema (V$SYSSTAT) y estadísticas de SQL:

1. V$SQLAREA,
2. V$SQL o V$SQLSTATS,
3. V$SQLTEXT,
4. V$SQL_PLAN, y
5. V$SQL_PLAN_STATISTICS.

Utilizando los datos recopilados en el paso dos, identifique las declaraciones SQL que utilizan la mayor cantidad de recursos. Una buena forma de identificar sentencias SQL candidatas es consultar V$SQLSTATS. V$SQLSTATS contiene información sobre el uso de recursos para todas las declaraciones SQL en el grupo compartido. Los datos en V$SQLSTATS deben ordenarse por uso de recursos. Los recursos más comunes son:

Los recursos más comunes son:

El búfer obtiene (V$SQLSTATS.BUFFER_GETS, para declaraciones de uso elevado de CPU)

Lecturas de disco (V$SQLSTATS.DISK_READS, para declaraciones de E/S elevadas)

Ordenaciones (V$SQLSTATS.SORTS, para muchas clases)

Un método para identificar qué declaraciones SQL están creando la carga más alta es comparar los recursos utilizados por una declaración SQL con la cantidad total de ese recurso utilizado en el período.

Para BUFFER_GETS, divida los BUFFER_GETS de cada instrucción SQL por el número total de búfer obtenidos durante el período. El número total de búfer que ingresa al sistema está disponible en la tabla V$SYSSTAT, para las lecturas lógicas de la sesión de estadísticas.

De manera similar, es posible distribuir el porcentaje de lecturas de disco que realiza una declaración del total de lecturas de disco realizadas por el sistema dividiendo V$SQL_STATS.DISK_READS por el valor de las lecturas físicas de la estadística V$SYSSTAT. Las secciones SQL del informe Repositorio automático de cargas de trabajo incluyen estos datos, por lo que no es necesario realizar los cálculos de porcentaje manualmente.

Una vez que haya identificado las sentencias SQL candidatas, el siguiente paso es recopilar información para ello es necesario examinar las declaraciones y afinarlas.

Recopilar información de un SQL identificado

El proceso de ajuste comienza determinando la estructura de las tablas e índices subyacentes. La información recopilada incluye lo siguiente:

Texto SQL completo de V$SQLTEXT

Estructura de las tablas a las que se hace referencia en la declaración SQL, generalmente describiendo la tabla en SQL*Plus

Definiciones de cualquier índice (columnas, ordenación de columnas) y si los índices son únicos o no únicos

Recopilar información de un SQL identificado

Estadísticas del optimizador para los segmentos (incluido el número de filas de cada tabla, selectividad de las columnas del índice), incluida la fecha en la que los segmentos se analizaron por última vez.

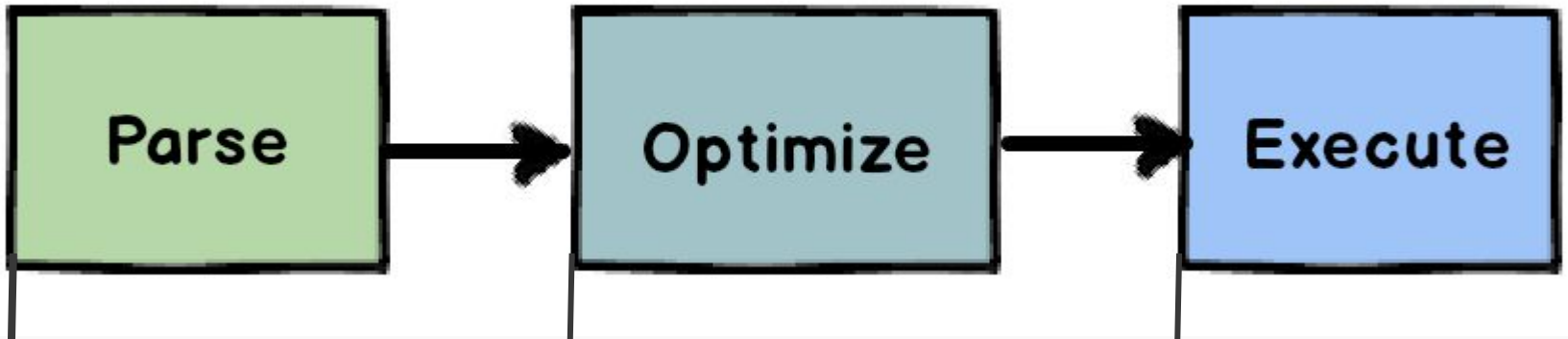
Definiciones de cualquier vista a la que se hace referencia en la declaración SQL

Repita los pasos dos, tres y cuatro para cualquier tabla a la que se haga referencia en las definiciones de vista que se encuentran en el paso cinco.

Plan optimizador para la declaración SQL (ya sea de EXPLAIN PLAN, V$SQL_PLAN o la salida TKPROF)

Cualquier plan optimizador anterior para esa declaración SQL

Ciclo de vida de una consulta

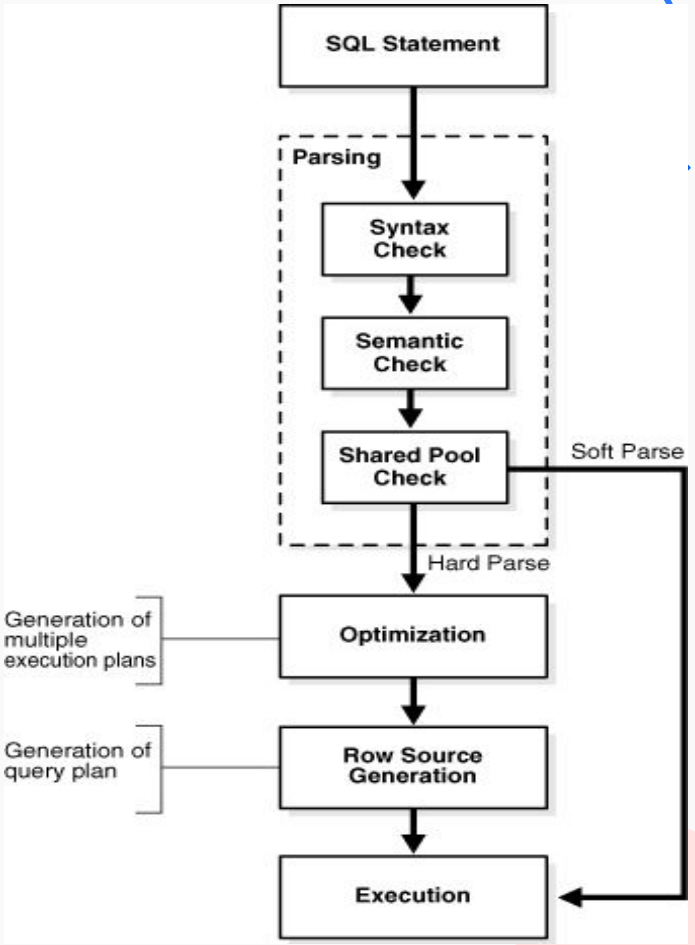


- Sintaxis
- Semántica
- Chequeo en el pool compartido

- Row source generator, genera el plan de ejecución

- Se ejecuta cada parte del plan (árbol de ejecución)

1. El **analizador de consultas** (SQL Parsing) se asegura de que la consulta sea **sintácticamente** correcta (por ejemplo, sin comas fuera de lugar) y **semánticamente** correcta (es decir, que las tablas existan), y regresa errores si no cumple. Si es correcta, entonces la convierte en una expresión algebraica y la pasa al siguiente paso.
2. El **planeador y optimizador de consultas** hace el trabajo pesado de pensar. Primero realiza optimizaciones directas. Después considera diferentes "planes de consulta" que pueden tener diferentes optimizaciones, estima el costo (CPU y tiempo) de cada consulta con base en el número de tuplas en las tablas relevantes, después escoge el plan óptimo y lo pasa al siguiente paso.
3. El **ejecutor de la consulta** toma el plan y lo convierte en operaciones de la base de datos, regresando los resultados si es que hay algunos.

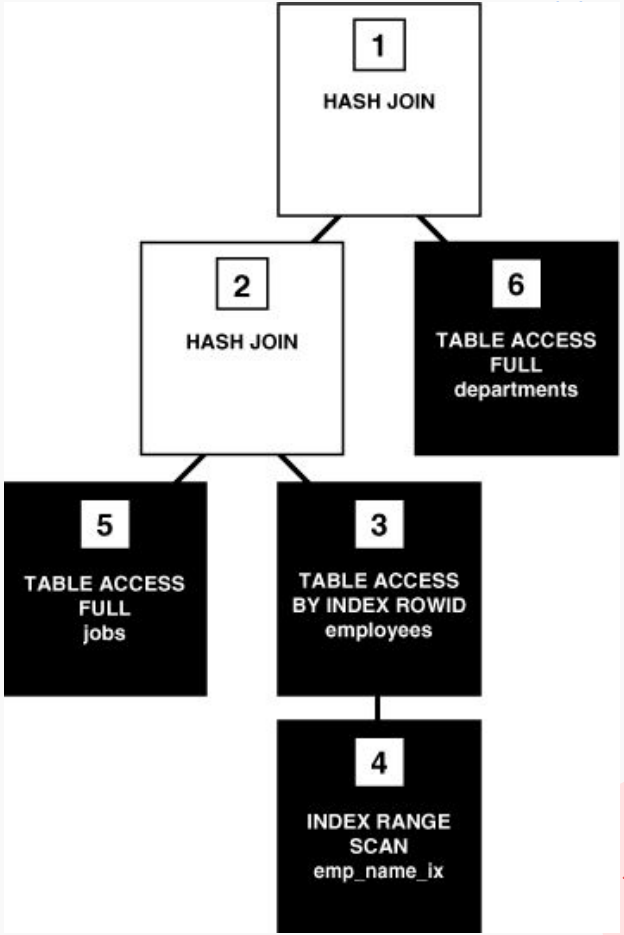


Plan de ejecución (Explain Plan)

Muestra el plan de ejecución elegido por el optimizador para sentencias SELECT, UPDATE, DELETE e INSERT.

Es la secuencia de operaciones que realiza el Gestor para correr la declaración.

El ‘core’ es el árbol de origen de filas (row source).



Plan de ejecución (Explain Plan)

Muestra la siguiente información:

- Un **orden** de las tablas a las que hace referencia la declaración.
- Un **método de acceso** para cada tabla mencionada en el comunicado.
- Un **método de unión** para tablas afectadas por operaciones de unión en la declaración
- **Operaciones** de datos cómo filtrar, ordenar o agregar

Además del árbol de origen de filas, la tabla del plan contiene información sobre lo siguiente:

- Optimización, como el costo y la cardinalidad de cada operación.
- Partición, como el conjunto de particiones a las que se accede.
- Ejecución paralela, como el método de distribución de entradas de unión.

Plan de ejecución (Explain Plan)

Los resultados de EXPLAIN PLAN permiten determinar si el optimizador selecciona un plan de ejecución particular, como por ejemplo, unión de bucles anidados. También le ayuda a comprender las decisiones del optimizador, como por qué el optimizador eligió una combinación de bucles anidados en lugar de una combinación hash, y le permite comprender el rendimiento de una consulta.

Plan de ejecución: Minimizando desechos

```
Rows       Execution Plan
--------   ---------------------------------------------------
      12   SORT AGGREGATE
       2    SORT GROUP BY
   76563     NESTED LOOPS
   76575      NESTED LOOPS
      19       TABLE ACCESS FULL CN_PAYRUNS_ALL
   76570       TABLE ACCESS BY INDEX ROWID CN_POSTING_DETAILS_ALL
   76570        INDEX RANGE SCAN (object id 178321)
   76563      TABLE ACCESS BY INDEX ROWID CN_PAYMENT_WORKSHEETS_ALL
11432983       INDEX RANGE SCAN (object id 186024)
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
COUNT(*)>0				
SORT		GROUP BY	34	11
VIEW	SYS.VM_NWVW_1		665	10
HASH		GROUP BY	665	10
HASH JOIN			665	9
Access Predicates				
C.CUSTOMER_ID=O.CUSTOMER_ID				
TABLE ACCESS	CUSTOMERS	FULL	320	3
MERGE JOIN		OUTER	665	6
TABLE ACCESS	ORDERS	BY INDEX ROWID	105	2
INDEX	SYS_C0013184	FULL SCAN	105	1
SORT		JOIN	665	4
Access Predicates				
ORDER_ID=OI.ORDER_ID(+)				
Filter Predicates				
ORDER_ID=OI.ORDER_ID(+)				
TABLE ACCESS	ORDER_ITEMS	FULL	665	3

- **Cardinality**– Estimate of the number of rows coming out of each of the operations.
- **Access method** – The way in which the data is being accessed, via either a table scan or index access.
- **Join method** – The method (e.g., hash, sort-merge, etc.) used to join tables with each other.
- **Join type** – The type of join (e.g., outer, anti, semi, etc.).
- **Join order** – The order in which the tables are joined to each other.
- **Partition pruning** – Are only the necessary partitions being accessed to answer the query?
- **Parallel Execution** – In case of parallel execution, is each operation in the plan being conducted in parallel? Is the right data redistribution method being used?

Access Methods

- Full scan table
- Table access by rowid
- Index unique scan
- Index range scan
- Index join

Join Methods

- Hash Join
- Nested loops joins
- Sort merge joins
- Cartesian join

https://docs.oracle.com/en/database/oracle/oracle-database/21/tgsql/optimizer-access-paths.html#GUID-CDC8B946-2375-4E5F-B50E-DE1E79EAE4CD

Plan de ejecución: Minimizando desechos

- Escaneos completos
- Escaneos de rango no selectivos
- Filtros de predicados tardíos
- Orden de unión incorrecta
- Operaciones de filtrado tardías