



OPTIMIZACIÓN DE CONSULTAS

PERFORMANCE EN LA BD

- Siempre que se tenga que realizar tareas de desarrollo y administración sobre una base de datos, deben considerarse las metodologías de afinamiento (tunning) para lograr un máximo rendimiento de la misma.

- Deben considerarse las características del diseño físico al igual que las de la aplicación (desarrollo de consultas) para optimizar las consultas y utilizar adecuadamente los recursos de procesamiento interno

AFINAMIENTO (TUNNING) DE BASES DE DATOS

■ OBJETIVOS

- Conseguir que las aplicaciones se ejecuten más rápido**
- Disminuir el tiempo de respuesta de las consultas/transacciones**
- Mejorar el rendimiento total de las transacciones**

AJUSTES DE BASES DE DATOS

- **ALGUNOS FACTORES A CARGO DEL ADMINISTRADOR DE LA BD**
 - Reducir los bloqueos debido a una excesiva concurrencia
 - Optimización del tamaño de los buffers y planificar los procesos
 - Considerar factores como el uso eficiente de memoria RAM y el acceso al disco

AJUSTES DE BASES DE DATOS

- *FACTORES A CARGO DEL DISEÑADOR DE LA BD*
 - **Selección adecuada de los índices**
 - **Ajustes en el diseño de la BD**
 - Desnormalización
 - Campos acumulados
 - Separación de datos históricos
 - **Ajuste de consultas**

OPTIMIZACIÓN DE CONSULTAS

Desde el rol del diseñador de la base de datos y del desarrollador, es importante afinar las sentencias de SQL:

- Determinando una estructura correcta de las tablas
- Definiendo los índices necesarios para las consultas frecuentes
- Tratando de utilizar PL/SQL en cuanto sea posible

Recursos críticos

- ❖ El tiempo de respuesta se define como el tiempo de servicio más el tiempo de espera requeridos para completar una tarea
- ❖ Depende de:
 - Cuántos clientes necesitan el recurso
 - Cuánto deben aguardar para utilizarlo
 - Cuánto tiempo retienen el recurso
- ❖ A mayor demanda de un recurso, la cola que se forma para utilizarlo crece exponencialmente, por tanto, habrá que limitar la demanda, o aumentar (u optimizar) los recursos

LA REGLA DEL 80/20

- Se podría decir que típicamente, el 20% de las transacciones (incluyendo sentencias SELECT), constituyen el 80% de la utilización del sistema. Por lo tanto, es a este 20% al que haya que dedicar.
- Se puede decir también que 50% de la utilización de los recursos (SO/BD) viene del 5% de las transacciones.

TÉCNICAS DE OPTIMIZACIÓN

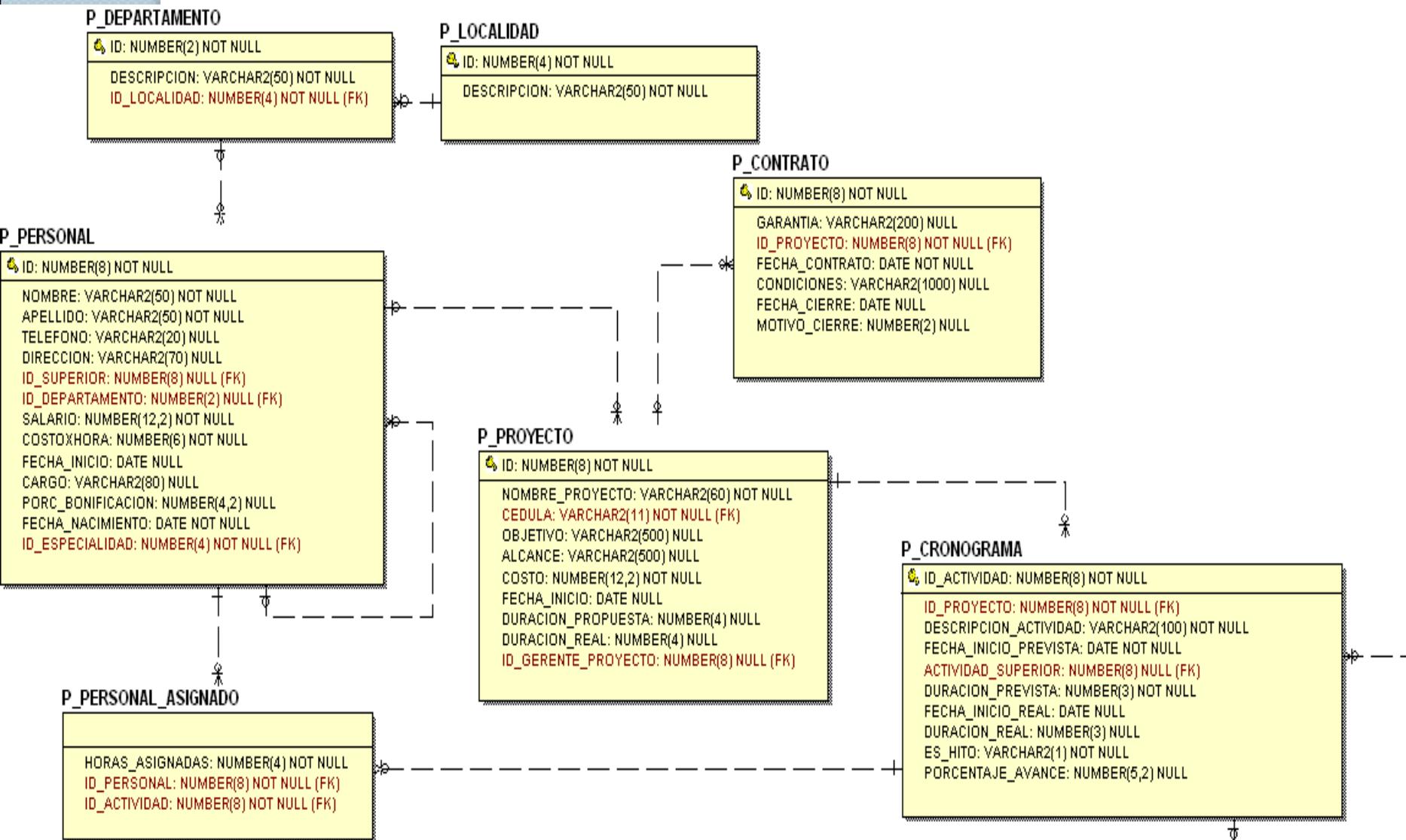
- Utilización de reglas heurísticas
- Uso de selectividad y estimaciones de costo

Reglas Heurísticas

Son técnicas de optimización que intentan modificar la consulta (utilizando una estructura de árbol) a los efectos de mejorar el rendimiento y generando planes de ejecución.

Ejemplo: Seleccionar el número de proyecto, el número de departamento, el apellido del jefe del departamento, su dirección y fecha de nacimiento de todos los proyectos de Asunción

Heurísticas básicas en la optimización de consultas



Heurísticas básicas en la optimización de consultas. Ej:

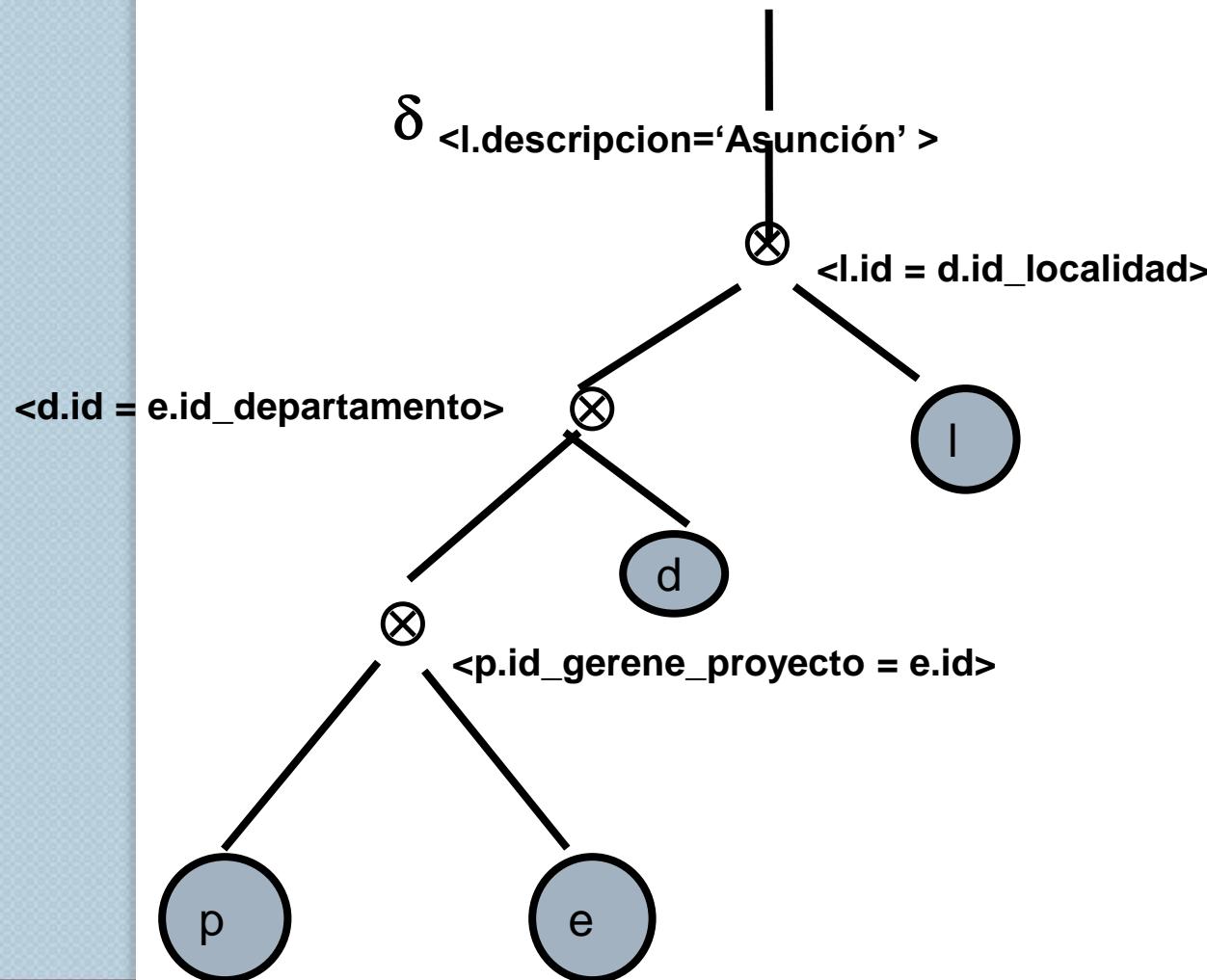
Seleccione todos los proyectos, con el nombre y apellido de sus gerentes localizados en Asunción

```
SELECT p.id, p.nombre_proyecto, e.nombre, e.apellido  
FROM p_proyecto p, p_personal e, p_departamento d,  
p_localidad l  
WHERE e.id = p.id_gerente_proyecto  
AND d.id = e.id_departamento  
AND l.id = d.id_localidad  
AND l.descripcion = 'Asunción';
```

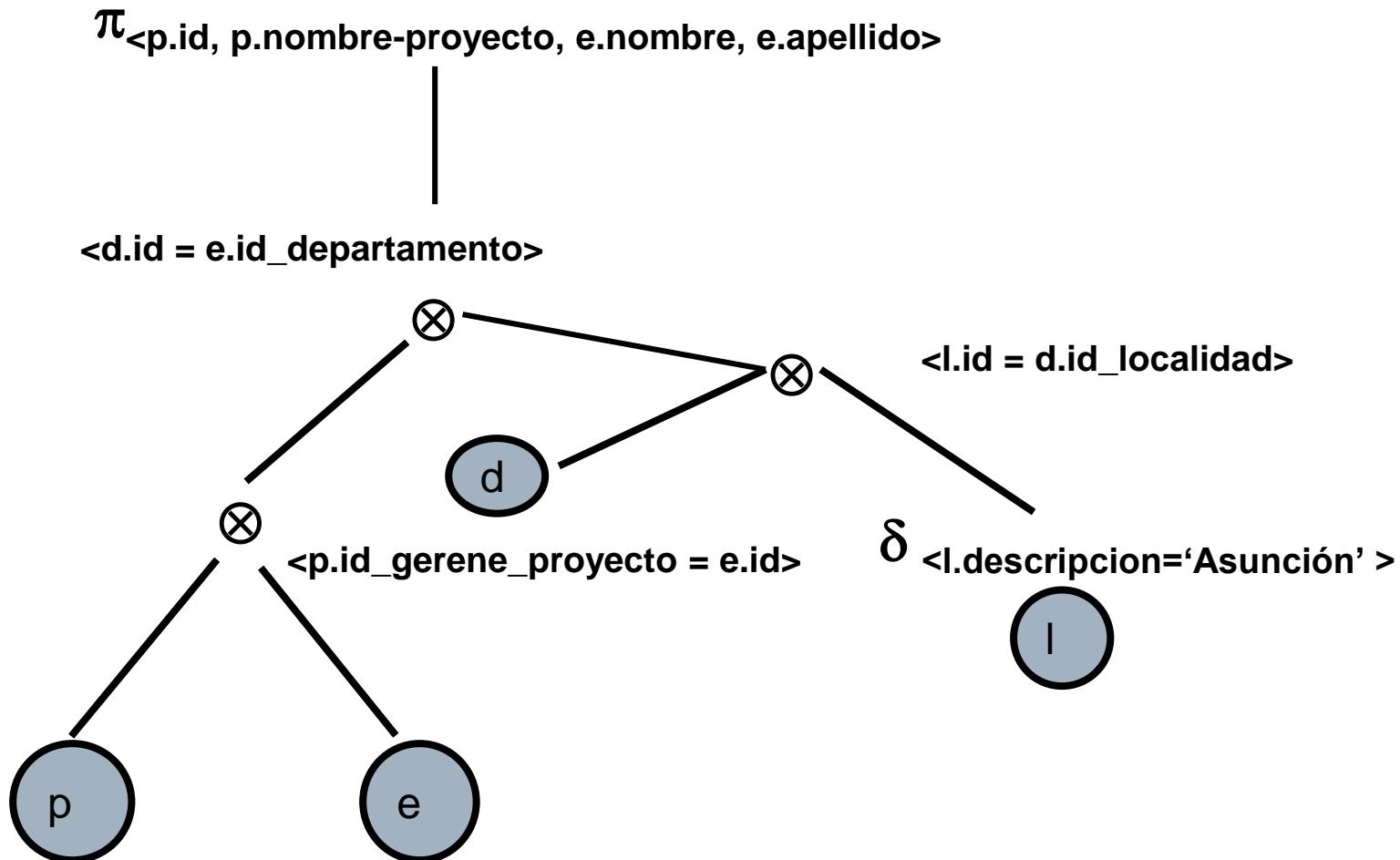
$$\pi_{<\text{id}, \text{nombre_proyecto}, \text{nombre}, \text{apellido}>} (\\delta_{<\text{descripcion}=\text{'Asunción'}>} (((\text{P_PROYECTO} \otimes_{<\text{id_gerente} = \text{id}>} \text{P_PERSONAL}) \otimes_{<\text{id_departamento} = \text{id}>} \text{P_DEPARTAMENTO}) \\otimes_{<\text{id_localidad} = \text{id}>} \text{P_LOCALIDAD}))$$

Heurísticas básicas en la optimización de consultas

$\pi_{<p.id, p.nombre-proyecto, e.nombre, e.apellido>}$



Heurísticas básicas en la optimización de consultas



Heurísticas básicas en la optimización de consultas

- Aplicar primero las operaciones que reducen los resultados intermedios, ejecutando las operaciones de SELECCIÓN lo más antes posible a fin de reducir el número de tuplas y la PROYECCIÓN a fin de reducir el número de atributos. Deberá por tanto desplazarse ambas operaciones en la parte inferior del árbol.

Reemplazar expresiones : $\sigma_{c1 \wedge c2 \wedge c12} (T1 \bowtie T2) \Rightarrow \sigma_{c12} (\sigma_{c1} (T1 \bowtie \sigma_{c2} (T2)))$

Optimización de consultas basadas en costo

- Un optimizador de consultas no puede depender exclusivamente de reglas heurísticas, pues debe considerar los costos de ejecución y considerar el más bajo
- Las funciones de costo son estimaciones y no funciones exactas. Se basa en información del catálogo del SGBD

Optimización de consultas basadas en costo: COMPONENTES

- Costo de acceso al almacenamiento secundario
- Costo de almacenamiento
- Costo de cómputo
- Costo de uso de memoria
- Costo de comunicación
- Estadísticas recolectadas del sistema (I/O, CPU y otros)

Optimización de consultas basadas en costo:

Elementos del catálogo (diccionario) de la BD

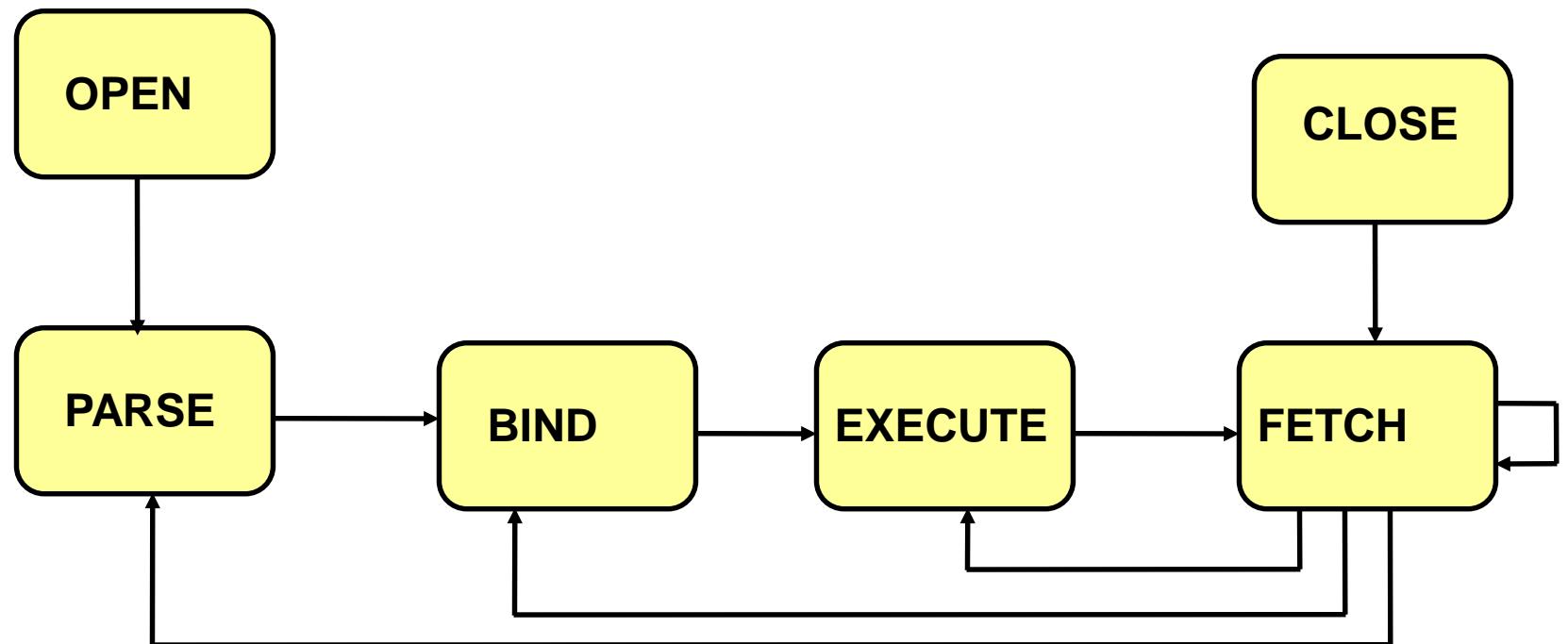
- Tamaño del archivo
- Número de registros
- Número de bloques
- Factor de bloqueo
- Método de acceso primario y sus atributos
- Número de valores distintos de un atributo y su selectividad a fin de determinar la cardinalidad de selección.

Procesamiento de Consultas

Procesamiento de consultas

- Una consulta SQL se descompone en bloques de consulta o unidades básicas que puedan traducirse
- Se traduce a una expresión del álgebra relacional
- El optimizador de consultas elige un plan de ejecución para cada bloque.

Procesamiento de sentencias



PARSE (Análisis)

- Durante este proceso se chequea la validez sintáctica y semántica de la sentencia, determina si el proceso tiene privilegios para ejecutar la sentencia y asigna un área privada para la sentencia

PARSE

- Si ya existe una representación de la misma sentencia analizada en el caché, la sentencia se ejecuta de inmediato. Si no, se genera la representación analizada, y se asigna un área compartida en el library cache para almacenarla. Este último proceso consume muchos más recursos.
- POR ESO TRATE DE ESCRIBIR LAS SENTENCIAS EXACTAMENTE IGUALES

Parse - Control de los cursosres compartidos

- ❖ Es posible modificar el parámetro que permite compartir los cursosres analizados.
- ❖ Este parámetro puede ser configurado a los siguientes valores
 - EXACT
 - SIMILAR
 - FORCE

EJ:

ALTER SESSION SET CURSOR_SHARING = 'EXACT'

BIND (Enlace)

- Chequea la sentencia para determinar si existen variables de enlace o acoplamiento
- Asigna valores a cada variable

EXECUTE y FETCH

- EXECUTE: Se usa el Plan de Ejecución para identificar las filas de datos del buffer de datos.
- FETCH: En esta fase se recupera las filas. Oracle para recuperar datos utiliza un array. Es posible configurar este arrayszie. El valor por defecto es 15 (15 filas).

EL OPTIMIZADOR

- El optimizador crea el plan de ejecución para la sentencia.
- El plan de ejecución consiste en una serie de operaciones que son realizadas en secuencia para ejecutar la sentencia

Reglas del optimizador

- Si la consulta NO tiene **where**, se realiza una *exploración completa*
- La utilización del índice se da de la siguiente manera
 - El índice no es utilizado cuando la **columna indexada** está **dentro de una expresión o en una función SQL**
 - El índice no es utilizado cuando la **columna indexada** es **comparada con nulo**

Reglas del optimizador

- El índice no es utilizado cuando la **columna indexada es comparada por diferencia**
- En comparaciones LIKE, el índice sólo es utilizado cuando el 1º carácter de la izquierda es distinto de %
- Un **índice compuesto** es utilizado si la primera columna del índice está declarada
- En caso de subconsultas, la tabla directriz es la de la subsentencia

Reglas del optimizador de Oracle

- En el caso de **AND**, Oracle utiliza hasta 5 índices concurrentes. Si entre ellos hay un índice único, sólo este será utilizado.
- En el caso de **OR**, el optimizador descompone la sentencia en tantas partes como condiciones existan separadas por este operador. Es suficiente que una condición no disponga de índice para que el optimizador no utilice ningún índice

Por lo tanto:

- Use equijoins cuando sea posible (uso de AND y el signo =)
- Evite uso de columnas transformadas en la cláusula WHERE
- Considere las conversiones implícitas, ya que esta tarea no permite que el optimizador determine una cardinalidad y selectividad válidas

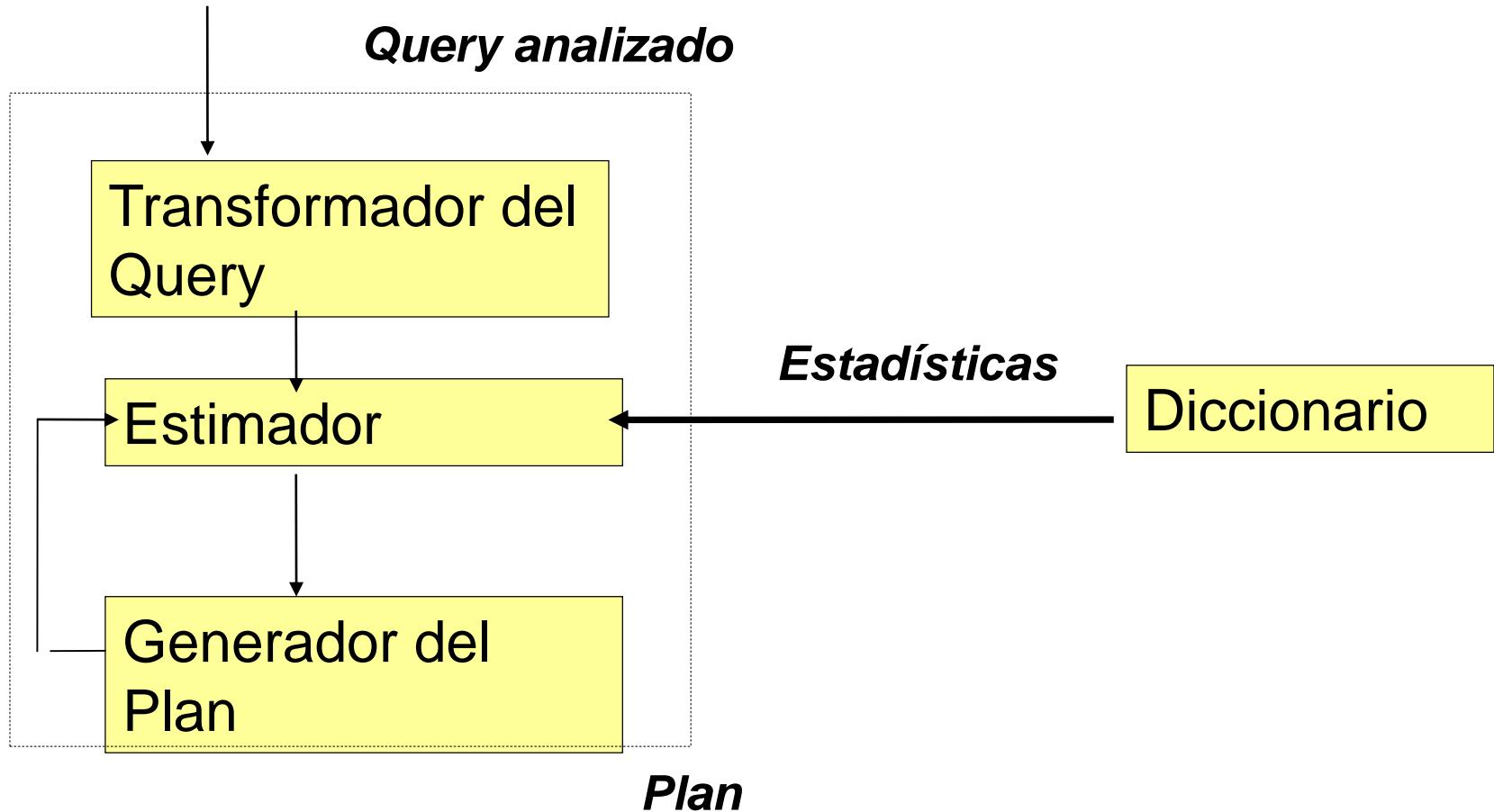
Otras recomendaciones

- Escriba sentencias SQL separadas para tareas específicas
- Uso del EXISTS vs IN: EXISTS es beneficioso cuando existe mayor selectividad en el query
- Puede modificarse la elección del Optimizador con el uso de HINTS.
- Puede reordenar los índices (use como primera columna aquella que es más frecuentemente utilizada en las cláusulas WHERE).
- Utilice el tipo de índice apropiado

Otras recomendaciones

- Se debe tener una clave primaria, con un índice único por ella.
- Todas las llaves foráneas deben estar indexadas.
- No se debe crear índices que están contenidos al inicio de otros.
- Nuevos índices con la cantidad mínima de columnas que los hagan selectivos
- Indexar por columnas not NULL.
- Poner al comienzo columnas más selectivas

Funciones del Optimizador



Afinamiento de sentencias SQL

- Crear la tabla PLAN_TABLE
- Usar EXPLAIN_PLAN para mirar cómo será procesada la sentencia
- Utilizar la utilidad AUTOTRACE para mostrar el plan de ejecución y las estadísticas

```
CREATE TABLE plan_table (
    STATEMENT_ID                               VARCHAR2(30)
    ,TIMESTAMP                                 DATE
    ,REMARKS                                   VARCHAR2(80)
    ,OPERATION                                  VARCHAR2(30)
    ,OPTIONS                                    VARCHAR2(30)
    ,OBJECT_NODE                                VARCHAR2(128)
    ,OBJECT_OWNER                               VARCHAR2(30)
    ,OBJECT_NAME                                VARCHAR2(30)
    ,OBJECT_INSTANCE                           NUMERIC
    ,OBJECT_TYPE                                VARCHAR2(30)
    ,OPTIMIZER                                  VARCHAR2(255)
    ,SEARCH_COLUMNS                            NUMBER
    ,ID                                         NUMERIC
    ,PARENT_ID                                 NUMERIC
    ,POSITION                                   NUMERIC
    ,COST                                       NUMERIC
    ,CARDINALITY                               NUMERIC
    ,BYTES                                      NUMERIC
    ,OTHER_TAG                                 VARCHAR2(255)
    ,PARTITION_START                           VARCHAR2(255)
    ,PARTITION_STOP                            VARCHAR2(255)
    ,PARTITION_ID                              NUMERIC
    ,OTHER                                     LONG
    ,DISTRIBUTION                            VARCHAR2(30)
) ;
```

SIGNIFICADO DE LAS COLUMNAS

Columna	Descripción
STATEMENT_ID	Es el nombre que se le da para las referencias futuras
OPERATION	El nombre de la operación SQL que se ejecuta en un paso dado
OPTIONS	Opciones para la operación
OBJECT_NAME	El nombre del objeto al que se refiere esta operación
ID	El número de paso en el plan de ejecución
PARENT_ID	El ID de la sentencia que es ‘padre’ de la sentencia actual
POSITION	El orden en el cual se ejecutan los pasos del mismo padre

El comando EXPLAIN PLAN

```
EXPLAIN PLAN  
[SET STATEMENT_ID = 'nombre_id']  
[INTO mi_plan_table]  
FOR sentencia_sql
```

El comando EXPLAIN PLAN (Ejemplo)

```
EXPLAIN PLAN
  SET STATEMENT_ID = 'DEMO02'
  FOR
    SELECT L.NOMBRE, P.NOMBRE || ','
          || P.APELLIDO "NOMBRE Y APELLIDO",
          P.DIRECCION,
          P.TELEFONO
        FROM B_PERSONAS P LEFT OUTER
      JOIN B_LOCALIDAD L
        ON L.ID = P.ID_LOCALIDAD;
```

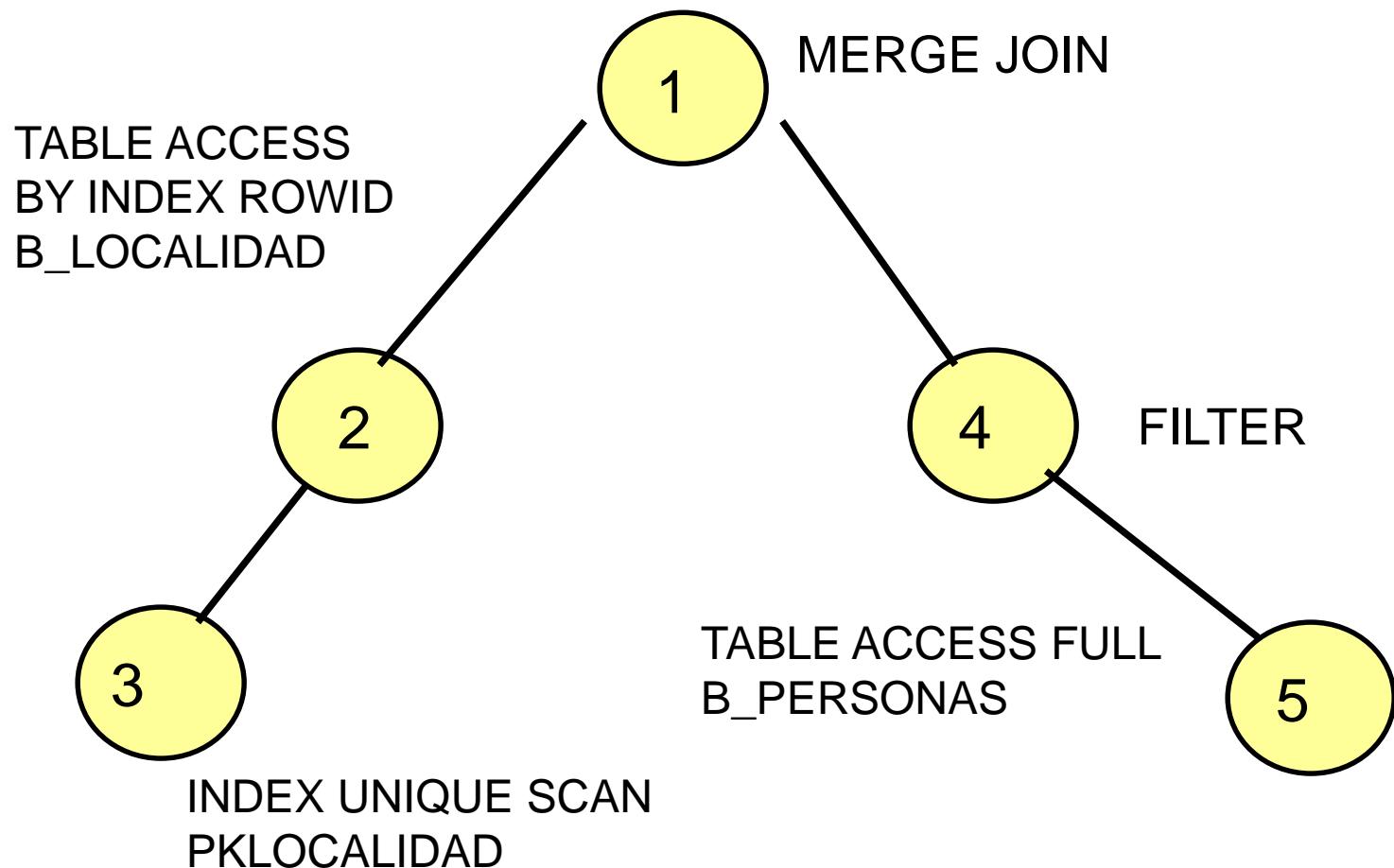
El Explain NO EJECUTA la sentencia SQL

Ver el *explain plan*

ID QUERY_PLAN

```
0      SELECT STATEMENT Costo =
1          MERGE JOIN
2              TABLE ACCESS BY INDEX ROWID B_LOCALIDAD
3                  INDEX UNIQUE SCAN PKLOCALIDAD
4                  FILTER
5                      TABLE ACCESS FULL    B_PERSONAS
```

Interpretación



Agregando un índice sobre b_personas.id_localidad

```
0   SELECT STATEMENT Costo =
1       NESTED LOOPS
2           TABLE ACCESS BY INDEX ROWID B_LOCALIDAD
3               INDEX UNIQUE SCAN PKLOCALIDAD
4           TABLE ACCESS BY INDEX ROWID B_PERSONAS
5               INDEX RANGE SCAN IND_ID_LOCALIDAD
```

USO DE HINTS

SINTAXIS

`/*+ hint */`

`/*+ hint(argumento) */`

Ejemplo

```
SELECT /*+ index(emp_alias ix_emp)
*/ FROM emp emp_alias
```

Algunos HINTS

- **ALL_ROWS** : normalmente es utilizado para procesos por lotes o los sistemas de almacenamiento de datos. ALL_ROWS indica al optimizador que utilice el mínimo de recursos para que devuelva el resultado completo
- **FIRST_ROWS**: el objetivo del optimizador es devolver la primera línea de la consulta en el menor tiempo posible.
- **RULE**: indica al optimizador que únicamente determine el plan de ejecución utilizando reglas estrictas sin tener en cuenta del contexto (estadísticas y costes de acceso) u otros hints en la consulta.
- **CHOOSE**: toma en cuenta las estadísticas si es que existen y utiliza el optimizador basado en costes.

Algunos HINTS (cont)

- **INDEX** (table [index]): fuerza el uso del índice "index"
- **INDEX_ASC** (table [index]): Funcionalidad idéntica al INDEX, pero sólo ejecuta una búsqueda ascendente.
- **INDEX_DESC** (table [index]): Funcionalidad idéntica al INDEX, pero sólo ejecuta una búsqueda descendente.
- **INDEX_FFS** (table index): Ejecuta un rápido recorrido de índice completo en lugar de un recorrido de tabla.

EL AUTOTRACE

Permite obtener automáticamente el plan de ejecución y algunas estadísticas adicionales

A diferencia del EXPLAIN_PLAN el AUTOTRACE sí ejecuta la sentencia, a no ser que se ponga SET AUTOTRACE TRACEONLY.

EL AUTOTRACE

- **SET AUTOTRACE OFF** : No se genera el reporte del AUTOTRACE. Es la opción por defecto
- **SET AUTOTRACE ON EXPLAIN**: Muestra solo el plan de ejecución del optimizador
- **SET AUTOTRACE ON STATISTICS**: Muestra solo las estadísticas de la sentencia SQL ejecutada
- **SET AUTOTRACE ON**: El AUTOTRACE incluye tanto el plan de ejecución, como las estadísticas de ejecución de la sentencia SQL
- **SET AUTOTRACE TRACEONLY**: Igual que el AUTOTRACE ON, pero no muestra el resultado de la ejecución del SQL. Si se active la opción de estadística, entonces el SQL se ejecuta, pero no se muestra el resultado.
- **SHOW AUTOTRACE**: Muestra las opciones actuales del AUTOTRACE.

Cambiar el modo del optimizador y desplegar costo CPU

```
sho parameter opt;
```

```
Set lines 1000;
```

```
optimizer_mode = first_rows/ all_rows/
first_rows_1, first_rows_10 and
first_rows_100;
```

```
alter session set
```

```
optimizer_index_cost_adj = 50 (Valor mas
pequeño, utilizará índices)
```