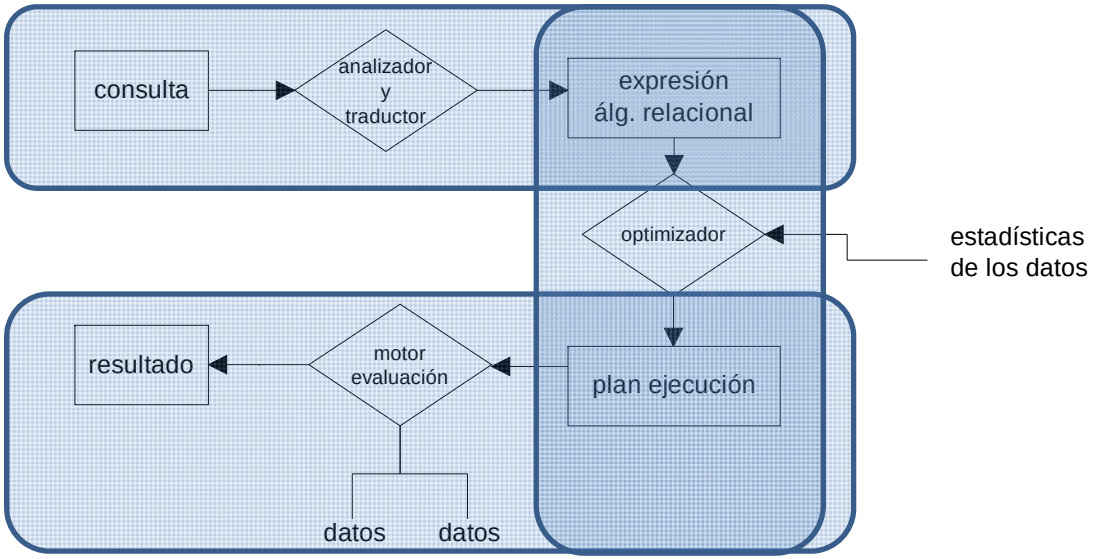
Tema.- Procesamiento y Optimización de consultas

Fundamentos de Bases de Datos (5º edic.). A. Silberschatz. McGraw-Hill [caps. 13-14]

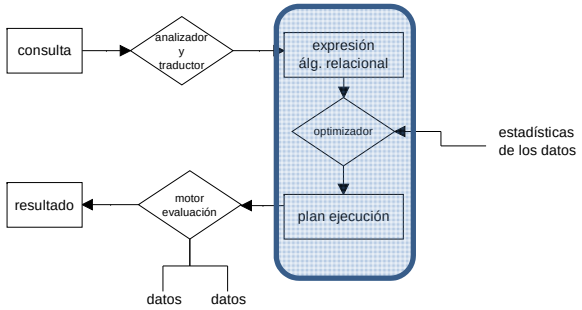
Procesamiento de consultas

- Pasos:



1. **Análisis y traducción**:
   - Comprobación de sintaxis, verificación de nombres
   - Construcción del árbol de consulta
   - Transformación a una expresión en álgebra relacional extendido
2. **Optimización**:
   - Indicar cómo evaluar la expresión en álgebra relacional (algoritmo a utilizar, índice a aplicar,…) ⇒ **primitiva de evaluación**
   - Determinar el **plan de ejecución** (= secuencia de primitivas de evaluación) que minimice el coste de ejecución de la consulta
3. **Evaluación**:
   - El motor de consultas ejecuta un plan de evaluación

Optimización de consultas



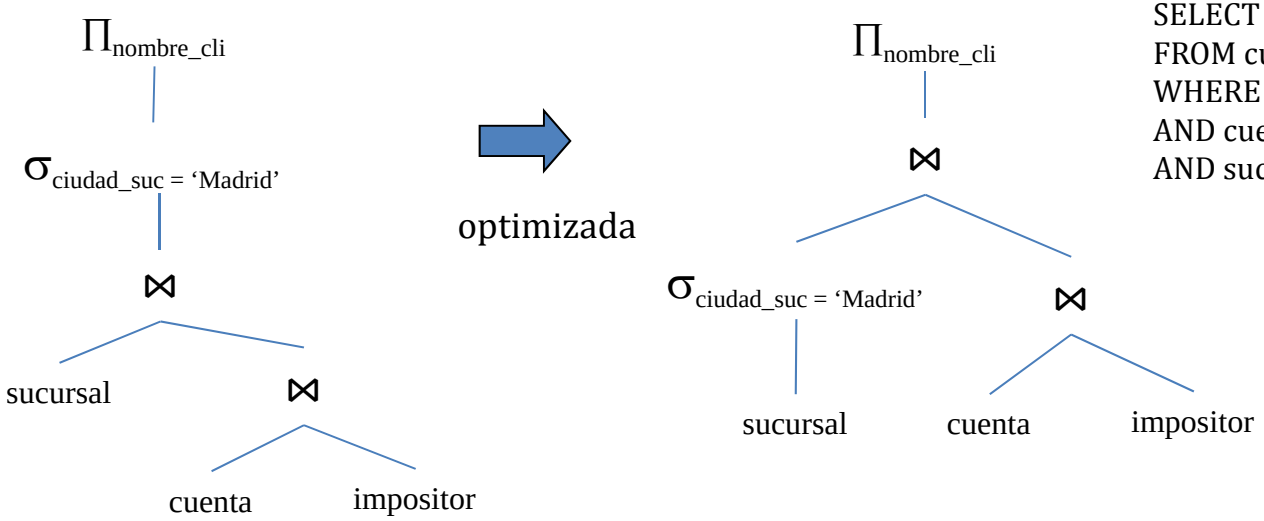
Nombres de todos los clientes con cuenta en una sucursal de Madrid

$\Pi_{nombre\_cli}\ (\sigma_{ciudad\_suc = \text{'Madrid'}}\ (sucursal \bowtie (cuenta \bowtie impositor)))$

optimizada

$\Pi_{nombre\_cli}\ (\sigma_{ciudad\_suc = \text{'Madrid'}}\ (sucursal) \bowtie (cuenta \bowtie impositor))$

sucursal (<u>nombre_suc</u>, ciudad_suc, activos)
cuenta (<u>num_cta</u>, nombre_suc, saldo)
impositor (<u>nombre_cli</u>, <u>num_cta</u>)
cliente (<u>nombre_cli</u>, edad, ciudad_cli, sueldo)




```
SELECT impositor.nombre_cli
FROM cuenta, impositor, sucursal
WHERE impositor.num_cta = cliente.num_cta
AND cuenta.nombre_suc = sucursal.nombre_suc
AND sucursal.ciudad_suc = "Madrid"
```

El optimizador diseña un plan de evaluación de consultas "óptimo" generando <u>planes alternativos</u>

1. genera expresiones equivalentes mediante ***reglas de equivalencia***
2. estima el COSTE de cada plan (utilizando inf. estadística de tablas, índices,etc.)
3. anota las expresiones alternativas para generar otros planes de ejecución

Reglas de equivalencia

- CASCADA DE σ => $\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$

  ej: $\sigma_{saldo>10 \wedge nombre\_suc=\text{“Madrid”}}(cuenta) = \sigma_{saldo>10}(\sigma_{nombre\_suc=\text{“Madrid”}}(cuenta))$

- CONMUTATIVIDAD DE σ => $\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$

  ej: $\sigma_{saldo>10}(\sigma_{nombre\_suc=\text{“Madrid”}}(cuenta)) = \sigma_{nombre\_suc = \text{“Madrid”}}(\sigma_{saldo>10}(cuenta))$

- CASCADA DE $\prod$ => $\prod_{L1}(\prod_{L2}(\ldots\prod_{Li}(E))\ldots)) = \prod_{L1}(E)$

  ej: $\prod_{num\_cta}(\prod_{num\_cta,\ nombre\_suc}(\prod_{num\_cta,\ nombre\_suc,\ saldo}(cuenta))) = \prod_{num\_cta}(cuenta)$

- $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$ $\sigma_{\theta 1}(E1 \bowtie_{\theta 2} E2) = E1 \bowtie_{\theta 1 \wedge \theta 2} E2$

  ej: $\sigma_{c.nombre\_suc = s.nombre\_suc}(cuenta \times sucursal) = cuenta \bowtie_{c.nombre\_suc = s.nombre\_suc} sucursal$

  ej: $\sigma_{saldo>activos}(cuenta \bowtie_{c.nombre\_suc = s.nombre\_suc} sucursal) =$

  $= cuenta \bowtie_{c.nombre\_suc = s.nombre\_suc \wedge saldo>activos} sucursal$

Reglas de equivalencia

- CONMUTATIVIDAD DE $\bowtie$ => $E1 \bowtie_{\theta} E2 = E2 \bowtie_{\theta} E1$

  ej: cuenta $\bowtie_{c.nombre\_suc = s.nombre\_suc}$ sucursal = sucursal $\bowtie_{c.nombre\_suc = s.nombre\_suc}$ cuenta

- ASOCIATIVIDAD DE $\bowtie$ => $(E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$

  ej: (cuenta $\bowtie$ sucursal) $\bowtie$ impositor = cuenta $\bowtie$ (sucursal $\bowtie$ impositor)

  $(E1 \bowtie_{\theta 1} E2) \bowtie_{\theta 2 \wedge \theta 3} E3 = E1 \bowtie_{\theta 1 \wedge \theta 3} (E2 \bowtie_{\theta 2} E3)$ donde $\theta 2$ implica solo atributos de E2 y E3

  ej: (cuenta$\bowtie_{c.num_cta=i.num_cta}$ impositor)$\bowtie_{i.nombre\_cli=c.nombre\_cli \wedge c.saldo > cl.sueldo}$ cliente=

  = cuenta$\bowtie_{c.num_cta=i.num_cta \wedge c.saldo > cl.sueldo}$ (impositor $\bowtie_{i.nombre\_cli=c.nombre\_cli}$ cliente)

- Desplazar σ hacia hojas en $\bowtie$ y X

  $\sigma_{\theta 0}(E1 \bowtie_{\theta} E2) = \sigma_{\theta 0}(E1) \bowtie_{\theta} E2$ donde $\theta 0$ implica solo atributos de E1

  $\sigma_{\theta 1 \wedge \theta 2}(E1 \bowtie_{\theta} E2) = \sigma_{\theta 1}(E1) \bowtie_{\theta} \sigma_{\theta 2}(E2)$ donde $\theta 1$ implica solo atributos de E1 y $\theta 2$ implica solo atributos de E2

  ej: $\sigma_{saldo>100}$(cuenta$\bowtie$sucursal) = $\sigma_{saldo>100}$ (cuenta)$\bowtie$sucursal

  $\sigma_{saldo>100 \wedge activos=200}$(cuenta $\bowtie$ sucursal) = $\sigma_{saldo>100}$(cuenta) $\bowtie \sigma_{activos=200}$ (sucursal)

Reglas de equivalencia

- Desplazar Π hacia las hojas en $\bowtie$ y X

  $\Pi_{L1 \cup L2}\ (E1 \bowtie_{\theta} E2) = \Pi_{L1}(E1) \bowtie_{\theta} \Pi_{L2}(E2)$ donde θ implica solo atributos de $L1 \cup L2$

  ej: $\Pi_{\text{nombre\_suc, activos}}\ (\text{sucursal} \bowtie \text{cuenta}) = \Pi_{\text{nombre\_suc, activos}}\ (\text{sucursal}) \bowtie \Pi_{\text{nombre\_suc}}\ (\text{cuenta})$

$\Pi_{L1 \cup L2}\ (E1 \bowtie_{\theta} E2) = \Pi_{L1 \cup L2}\ (\Pi_{\ L1 \cup L3}(E1) \bowtie_{\theta} \Pi_{\ L2 \cup L4}(E2))$

donde L1 y L2 son conjuntos de atributos de E1 y E2, respectivamente
L3 atributos de E1 que están implicados en θ pero no están en L1
L4 atributos de E2 que están implicados en θ pero no están en L2

ej: $\Pi_{\text{activos, saldo}}\ (\text{sucursal} \bowtie \text{cuenta}) =$
$\Pi_{\text{activos, saldo}}\ (\Pi_{\text{nombre\_suc, activos}}\ (\text{sucursal}) \bowtie \Pi_{\text{nombre\_suc, saldo}}(\text{cuenta}))$

Reglas de equivalencia

- Un conjunto de reglas es MÍNIMO si no se puede obtener ninguna regla a partir de la unión de otras.
- Los optimizadores utilizan conjuntos mínimos de reglas de equivalencia
- Los optimizadores generan sistemáticamente expresiones equivalentes a la consulta dada
- Una buena ordenación de los joins reduce el tamaño de los resultados

Ej:

$$\sigma_{ciudad\_suc = \text{'Madrid'}} (sucursal) \bowtie (cuenta \bowtie impositor)$$

Como es probable que:
1) (cuenta ⋈ impositor) de lugar a una relación muy grande (tantas tuplas como impositores existan)
2) el número de cuentas de las sucursales de Madrid es pequeño

sería mejor aplicar la regla de asociatividad del join, dando lugar a la expresión:

$$\mathbf{(\sigma_{ciudad\_suc = \text{'Madrid'}} (sucursal) \bowtie cuenta) \bowtie impositor}$$

Tipos de optimización: Heurística

- Reordena los componentes del árbol de consultas inicial para intentar reducir el coste de la optimización.

1. **Desplazar σ hacia las hojas del árbol**

   CASCADA DE σ => $\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$

   CONMUTATIVIDAD DE σ => $\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$

   $\sigma_{\theta 0}(E1 \bowtie_{\theta} E2) = \sigma_{\theta 0}(E1) \bowtie_{\theta} E2$ donde $\theta 0$ implica solo atributos de E1

   $\sigma_{\theta 1 \wedge \theta 2}(E1 \bowtie_{\theta} E2) = \sigma_{\theta 1}(E1) \bowtie_{\theta} \sigma_{\theta 2}(E2)$ donde $\theta 1$ implica solo atributos de E1 y $\theta 2$ implica solo atributos de E2

Tipos de optimización: Heurística

2. Sustituir el producto cartesiano seguido de σ por ⋈

$\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$

$\sigma_{\theta 1}(E1 \bowtie_{\theta 2} E2) = E1 \bowtie_{\theta 1 \wedge \theta 2} E2$

3. Determinar las operaciones σ y ⋈ que producen menos tuplas, y ejecutarlas cuanto antes

CONMUTATIVIDAD DE σ => $\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$

ASOCIATIVIDAD DE ⋈ => $(E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$

$(E1 \bowtie_{\theta 1} E2) \bowtie_{\theta 2 \wedge \theta 3} E3 = E1 \bowtie_{\theta 1 \wedge \theta 3} (E2 \bowtie_{\theta 2} E3)$

donde θ2 implica solo atributos de E2 y E3

4. Desplazar ∏ hacia las hojas del árbol

CASCADA DE ∏ => $\prod_{L1}(\prod_{L2}(\ldots\prod_{Li}(E))\ldots)) = \prod_{L1}(E)$

$\prod_{L1 \cup L2}(E1 \bowtie_{\theta} E2) = (\prod_{L1}(E1)) \bowtie_{\theta} (\prod_{L2}(E2))$

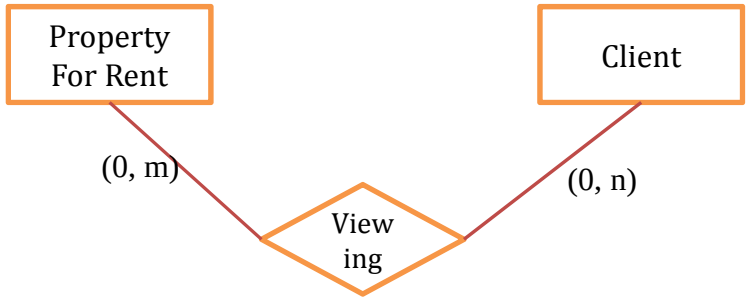
$\prod_{L1 \cup L2}(E1 \bowtie_{\theta} E2) = \prod_{L1 \cup L2}((\prod_{L1 \cup L3}(E1)) \bowtie_{\theta} (\prod_{L2 \cup L4}(E2)))$

Ejemplo utilización de las reglas de transformación

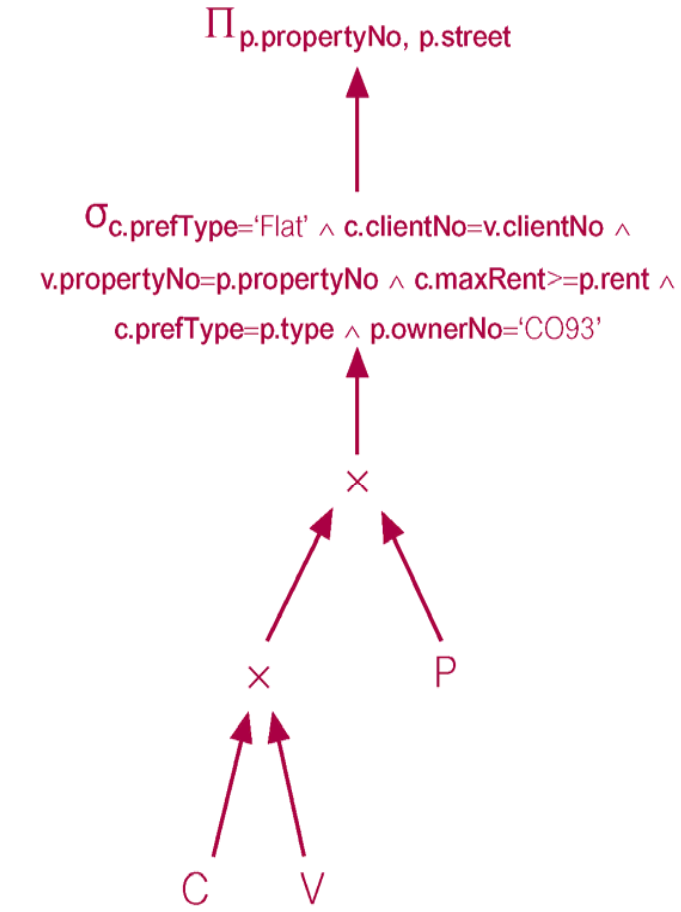
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)
Viewing (clientNo, propertyNo, comment, viewDate)

Para los inquilinos que estén buscando apartamentos, localizar los inmuebles que satisfacen sus requisitos y son propiedad del propietario CO93.

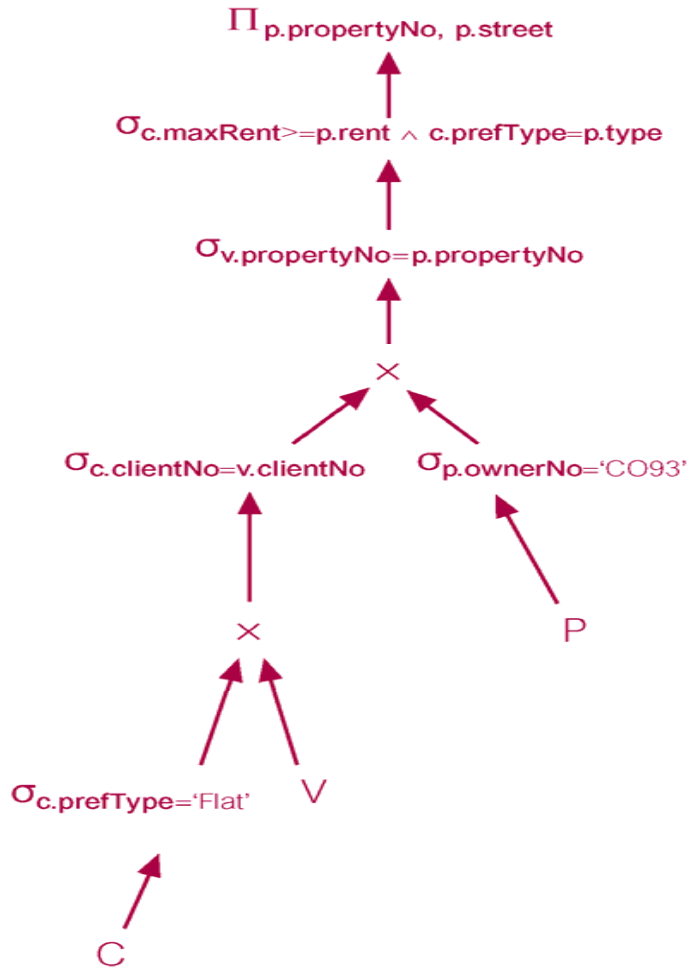
```
SELECT p.propertyNo, p.street
FROM Client c, Viewing v, PropertyForRent p
WHERE          c.prefType = 'Flat' AND
               c.clientNo = v.clientNo AND
               v.propertyNo = p.propertyNo AND
               c.maxRent >= p.rent AND
               c.prefType = p.type AND
               p.ownerNo = 'CO93';
```



Ejemplo utilización de las reglas de transformación

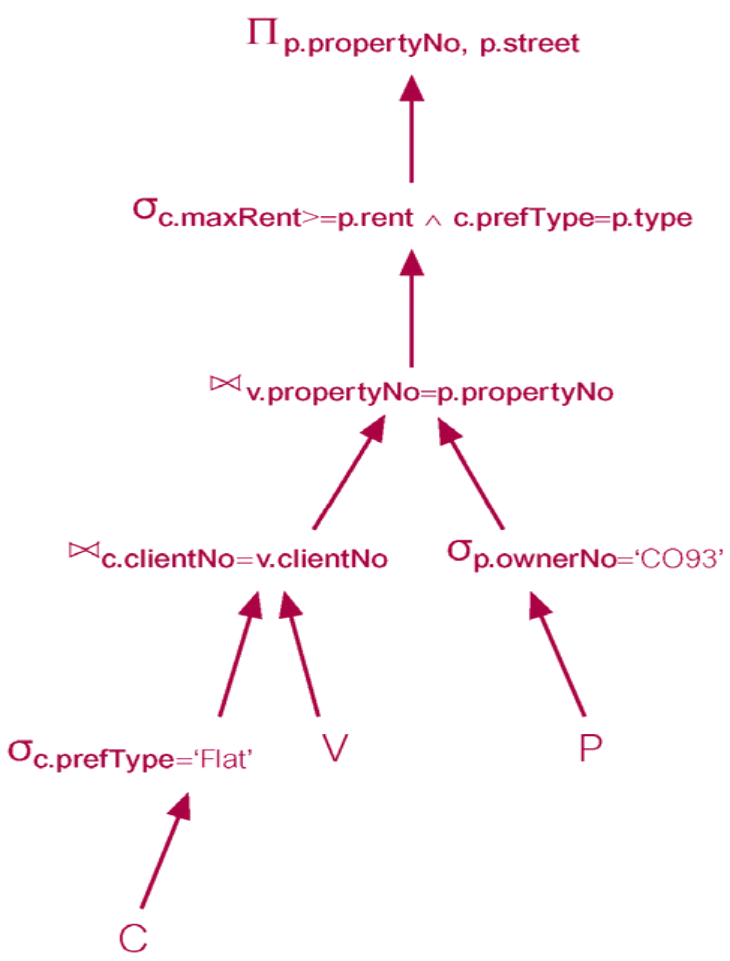


(a)

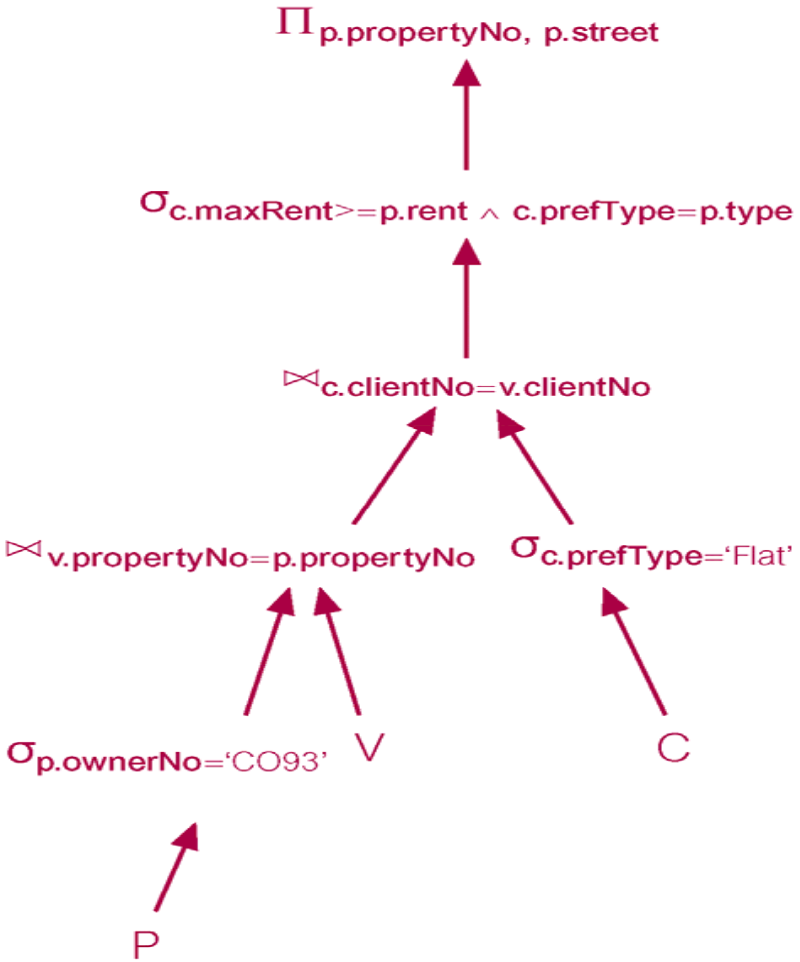


(b) Desplazar σ hacia las hojas del árbol

Ejemplo utilización de las reglas de transformación

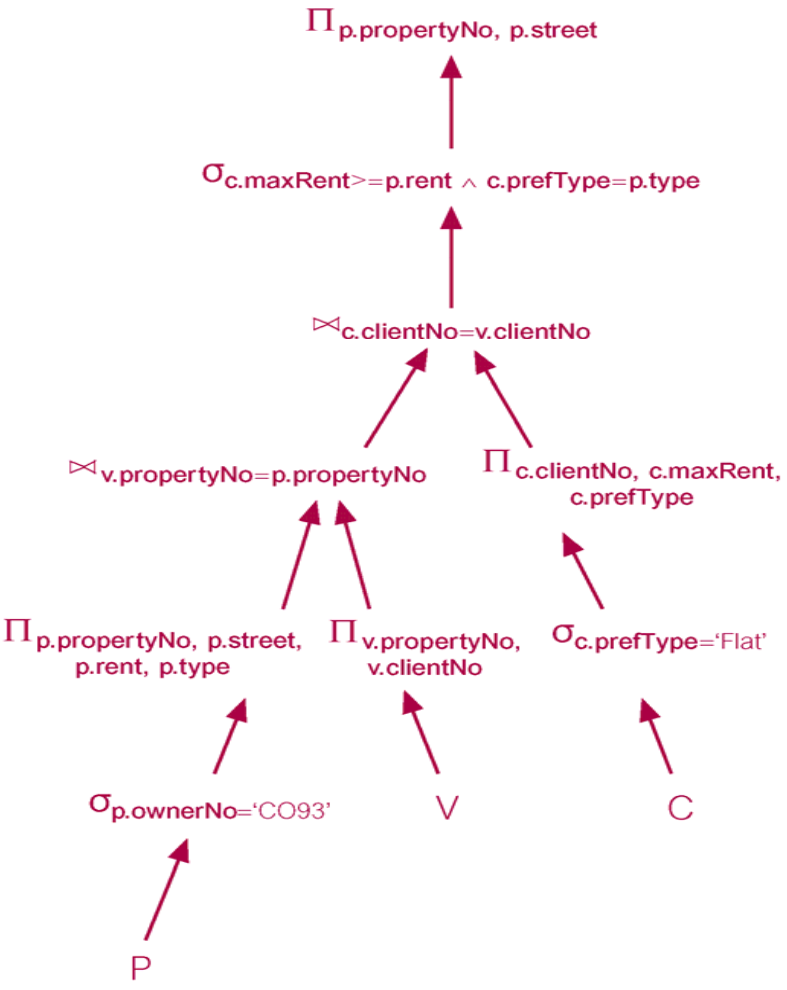


(c) Sustituir el producto cartesiano seguido de σ por ⋈

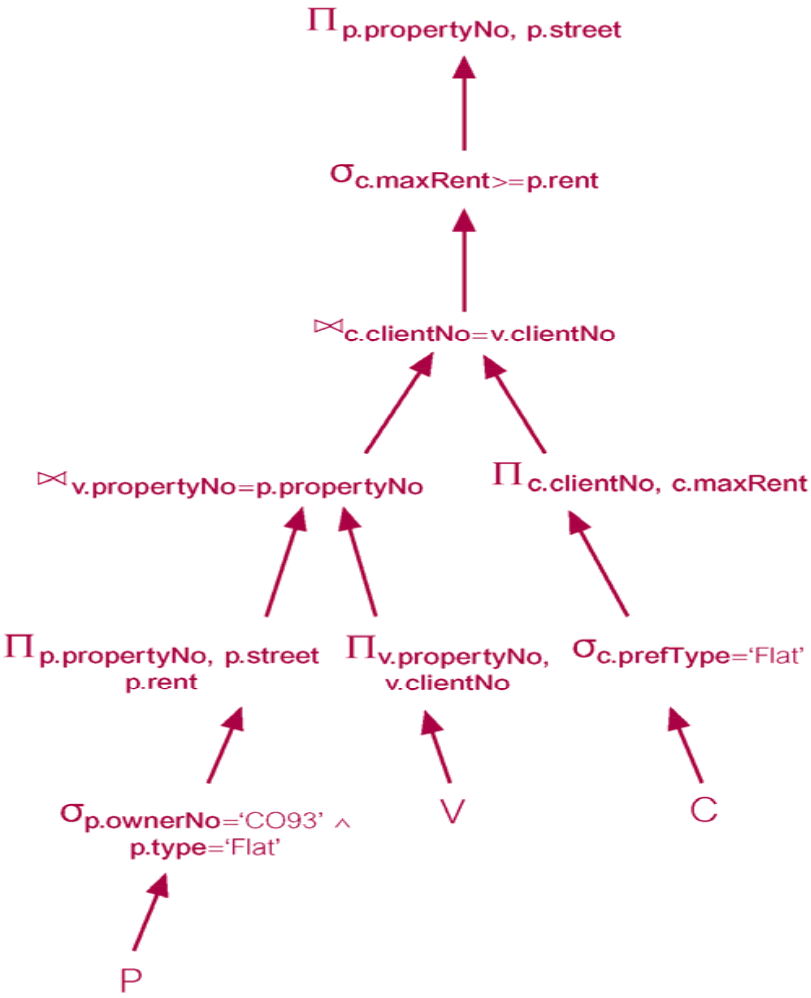


(d) Determinar las operaciones σ y ⋈ que producen menos tuplas, y ejecutarlas cuanto antes *(en el ejemplo, el nº de propiedades es inferior al nº de clientes)*

Ejemplo utilización de las reglas de transformación



(e) Desplazar Π hacia las hojas del árbol



(f) Sustitución de c.prefType = p.type por (p.type = 'Flat')

Tipos de optimización: basada en coste

- Se genera una gama de planes de evaluación empleando reglas de equivalencia
- Se realiza una estimación estadística de los resultados de las expresiones
  - Ver apartado 14.3 Silberstchatz, A.; Korth, H.; Sudarshan, S. Fundamentos de Bases de Datos. Mc Graw-Hill (5ª edición)
- Se escoge el plan de evaluación de coste mínimo.

Estimación estadística de tamaños

Información del catálogo:

- **n_r** (*nº registros de r*)
- t_r (*tamaño tupla de r*)

```
select    AVG(VSIZE(deptno)) + AVG(VSIZE(dname))
          + AVG(NVL(VSIZE(loc),0))
from      DEPT;
```

- f_r (*factor bloqueo de r*)

```
select    NUM_ROWS, BLOCKS, NUM_ROWS / BLOCKS
from      USER_TABLES
where     TABLE_NAME = 'DEPT';
```

- **V(A, r)** (*nº valores distintos para el atributo A en la relación r*)

Estimación estadística de tamaños

- Tamaño de la proyección de atributos

Π_A **(r)** => **V(A, r)** porque elimina duplicados

Ej: Π_{edad} *(empleado) donde el rango de edad es [20-60)*

=> V(edad, empleado) = 40 valores = 40 tuplas

- Tamaño del filtro de tuplas (igualdad)

$$\sigma_{A = v}(r) => n_r \frac{casos\ favorables}{casos\ posibles} = n_r \frac{1}{V(A,r)}$$

Ej: $\sigma_{ciudad\_suc = \text{“Madrid”}}(sucursal)$ *donde* $n_{sucursal} = 1000$ *y* $V(ciudad\_suc, r) = 5$ *equiprobables*

Probabilidad de que 1 tupla satisfaga la condición: $\frac{casos\ favorables}{casos\ posibles} = \frac{1}{5}$

Nº de tuplas que satisfacen la condición: $1000 \frac{1}{5} = \mathbf{200\ tuplas}$

Estimación estadística de tamaños

- Tamaño del filtro de tuplas (rango)

$\boldsymbol{\sigma_{A \le v}(r)} \Rightarrow \boldsymbol{n_r}\left(\frac{\boldsymbol{v - min(A,r)}}{\boldsymbol{max(A,r) - min(A,r)}}\right)$ *si se conoce v y hay distribución uniforme de valores*

Ej: $\sigma_{edad \le 30}$ *(empleado)* *edad [20-60), 2000 empleados*

Probabilidad de que 1 tupla satisfaga la condición: $\frac{casos\ favorables}{casos\ posibles} = \frac{30-20}{60-20} = \frac{1}{4}$

Nº de tuplas que satisfacen la condición: $2000\ \frac{1}{4} = \mathbf{500\ tuplas}$

$\boldsymbol{\sigma_{A \le v}(r)} \Rightarrow \frac{\boldsymbol{n_r}}{\boldsymbol{2}}$ *si NO se conoce v*

Ej: $\sigma_{edad \le \text{“X”}}$ *(empleado)* *edad [20-60), 2000 empleados*

Probabilidad de que 1 tupla satisfaga la condición: $\frac{1}{2}$

Nº de tuplas que satisfacen la condición: $2000\ \frac{1}{2} = \mathbf{1000\ tuplas}$

Estimación estadística de tamaños

- Tamaño del filtro de tuplas (rango)

$$\sigma_{A > v}(r) => n_r \left(\frac{max(A,r) - v}{max(A,r) - min(A,r)} \right)$$

Ej: $\sigma_{edad > 30}$ *(empleado)* *edad [20-60), 2000 empleados*

Probabilidad de que 1 tupla satisfaga la condición: $\frac{casos\ favorables}{casos\ posibles} = \frac{60-30}{60-20} = \frac{3}{4}$

Nº de tuplas que satisfacen la condición: $2000\ \frac{3}{4} = \mathbf{1500\ tuplas}$

Estimación estadística de tamaños

- Tamaño del filtro de tuplas (conjunción de condiciones)

$\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta p}$ (r)

Probabilidad de que 1 tupla satisfaga la condición θ1: $P_{\theta 1} \frac{casos\ favorables}{casos\ posibles}$

Si las condiciones son independientes, la probabilidad de que 1 tupla cumpla todas las condiciones es el producto de las probabilidades de las condiciones: $P_{\theta 1} * P_{\theta 2} * \ldots * P_{\theta p}$

Nº tuplas de la selección completa es: $\boldsymbol{n_r}(P_{\theta 1} * P_{\theta 2} * \ldots * P_{\theta p})$

Ej: σ *(edad < 30) ^ (salario = 120) (empleado) donde edad [20-60), 2000 empleados, salario [50, 150)*

Probabilidad de que 1 tupla satisfaga la condición (edad < 30): $\frac{casos\ favorables}{casos\ posibles} = \frac{1}{4}$

Probabilidad de que 1 tupla satisfaga la condición (salario = 120): $\frac{casos\ favorables}{casos\ posibles} = \frac{1}{150-50} = \frac{1}{100}$

Probabilidad de que 1 tupla satisfaga (edad < 30) ^ (salario = 120): $\frac{1}{4}\frac{1}{100} = \frac{1}{400}$

Nº de tuplas que satisface (edad < 30) ^ (salario = 120): $2000\ \frac{1}{400} = \mathbf{5\ tuplas}$

Estimación estadística de tamaños

- Tamaño del filtro de tuplas (disyunción de condiciones)

$\sigma_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta p}$ **(r)**

Probabilidad de que 1 tupla satisfaga la condición θ1: $P_{\theta 1}$ $\frac{casos\ favorables}{casos\ posibles}$

Si las condiciones son independientes, la probabilidad de que 1 tupla satisfaga la disyunción es:
(1 menos la probabilidad de que no satisfaga ninguna) $1-(1-P_{\theta 1}) * (1-P_{\theta 2}) * \ldots * (1-P_{\theta p})$

Nº tuplas de la selección completa es: $n_r [1-(1-P_{\theta 1}) * (1-P_{\theta 2}) * \ldots * (1-P_{\theta p})]$
]

Ej: σ *(edad < 30) v(salario = 120) (empleado) donde edad [20-60), 2000 empleados, salario [50, 150)*

Probabilidad de que 1 tupla satisfaga la condición (edad < 30): $\frac{casos\ favorables}{casos\ posibles} = \frac{1}{4}$

Probabilidad de que 1 tupla satisfaga la condición (salario = 120): $\frac{casos\ favorables}{casos\ posibles} = \frac{1}{150-50} = \frac{1}{100}$

Probabilidad de que 1 tupla satisfaga (edad < 30) ó (salario = 120): $1-\left(1-\frac{1}{4}\right) * \left(1-\frac{1}{100}\right) = \frac{103}{400}$

Nº de tuplas que satisface (edad < 30) ó (salario = 120): $2000\ \frac{103}{400} = \mathbf{515\ tuplas}$

Estimación estadística de tamaños

- Join de relaciones r ∞ s

a) Si R ⋂ S = 0 => $\boldsymbol{n_r * ns}$

b) Si R ⋂ S es clave de R => cada tupla de s se combina como máximo con 1 de R => el nº de tuplas del join NO es mayor que las tuplas de S.
   Si, además, los atributos comunes son clave foránea de S, entonces el nº de tuplas del join es EXACTAMENTE el nº de tuplas de S.

c) Si R ⋂ S NO es clave de R ni de S, suponiendo que todos los valores aparecen con igual probabilidad, entonces:
   a) cada tupla de R produce $\frac{\boldsymbol{n_s}}{\boldsymbol{V(A,s)}}$ tuplas en el join, donde A son los atributos comunes
      => todas las tuplas de R producen $\boldsymbol{n_r}\ \frac{\boldsymbol{n_s}}{\boldsymbol{V(A,s)}}$ tuplas
   b) cada tupla de S produce $\frac{\boldsymbol{n_r}}{\boldsymbol{V(A,r)}}$ tuplas en el join, donde A son los atributos comunes
      => todas las tuplas de S producen $\boldsymbol{n_s}\ \frac{\boldsymbol{n_r}}{\boldsymbol{V(A,r)}}$ tuplas
   c) como $V(A,r) \neq V(A,s)$, el nº de tuplas del join es de R produce mi$\boldsymbol{n}$ $(\frac{\boldsymbol{n_r}\ \boldsymbol{n_s}}{\boldsymbol{V(A,r)}}, \frac{\boldsymbol{n_r}\ \boldsymbol{n_s}}{\boldsymbol{V(A,s)}})$

Optimización de consultas en Oracle

- **Optimización basada en coste**

  Genera {planes de evaluación} aplicando reglas de equivalencia
  Elige el plan de coste mínimo
  PERO... nº planes puede ser grande

- Para cada sentencia SQL, el optimizador ejecuta las siguientes operaciones:
  1. Evaluación de expresiones y condiciones: El optimizador evalúa lo antes posible las expresiones y condiciones que contienen constantes
  2. Transformación de sentencias: Sentencias complejas que involucran sentencias correlacionadas o vistas, son transformadas en una expresión de join equivalente

```
select nombre_cliente
from prestatario p
where exists (select *
        from impositor i
        where i.nombre_cliente=p.nombre_cliente)
```

descorrelación

```
select nombre_cliente
from prestatario p, impositor i
where i.nombre_cliente=p.nombre_cliente
```

http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/optimops.htm

Optimización de consultas en Oracle

3. Elección de las metas de la optimización:
   1. **tasa de procesamiento**: minimización de la cantidad de recursos necesaria para procesar **todas las filas** a las que accede la sentencia, o
   2. **tiempo de respuesta**: minimizar la cantidad de recursos necesaria para procesar **la(s)** ***primera(s)*** **fila(s)** a la que accede la consulta.

   La configuración se establece mediante el parámetro OPTIMIZER_MODE, cuyos valores posibles son:
   - *ALL_ROWS*: Establece la optimización por tasa de procesamiento. Es la opción por defecto.
   - *FIRST_ROWS_n*: Minimiza la cantidad de recursos necesaria para procesar las *n* primeras filas a las que accede la consulta; *n* puede valer 1, 10, 100, o 1000.

4. Elección de los caminos de acceso: Para cada tabla accedida por una sentencia, el optimizador elige una o más rutas de acceso (uso de índices, rutas completas) para obtener sus datos.
5. Elección del orden de los joins: Para cada sentencia join que une más de dos tablas, el optimizador elige qué par de tablas se une en primer lugar, qué tabla se unirá con este resultado, y así sucesivamente.

http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/optimops.htm

Optimización de consultas en Oracle: Estadísticas

- El optimizador basado en costes depende para su funcionamiento de las estadísticas disponibles para todas las tablas, clústeres e índices a los que accede la consulta.
- Las estadísticas utilizadas por el optimizador se encuentran en el diccionario de datos y son mantenidas por el usuario.
  - El paquete PL/SQL DBMS_STATS gestiona las estadísticas de todos los objetos de la BD
  - Para recopilar las estadísticas de un esquema se utiliza:

    EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS ('Manager',
    DBMS_STATS.AUTO_SAMPLE_SIZE);

    El último parámetro le dice a Oracle que deber determinar el mejor tamaño muestral para la obtención de unas buenas estadísticas.
- Para mantener la efectividad del optimizador, deben tenerse estadísticas representativas de los datos
  - Para columnas de tablas que contengan valores con muchas variaciones en el número de duplicados (datos sesgados), deben utilizarse **histogramas**.

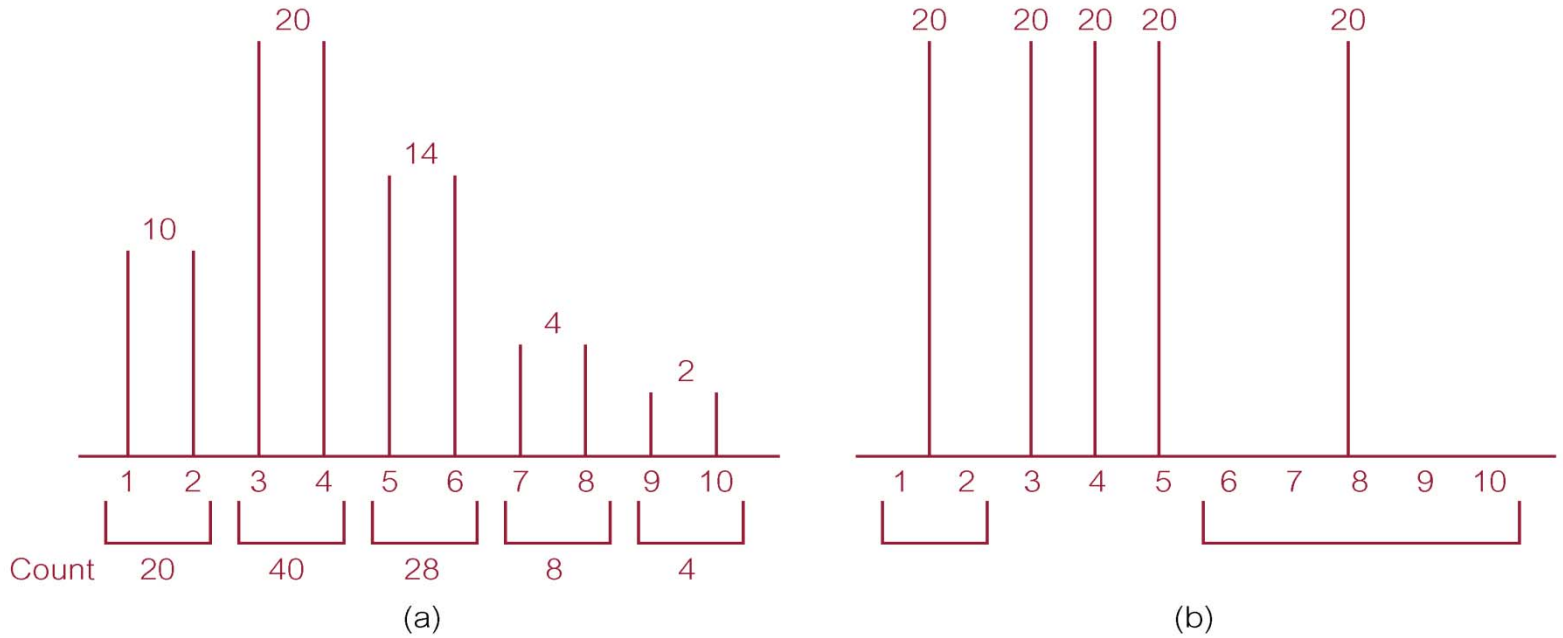
http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/optimops.htm

Optimización de consultas en Oracle: Histogramas

- Un histograma es una estructura de datos que puede utilizarse para mejorar la estimación del número de tuplas en un resultado.

- Hay dos tipos de histogramas:
  - *Histograma con anchuras equilibradas*, que divide los datos en un número fijo (denominados ranuras) y cada uno de los cuales contiene un recuento del número de valores que caen dentro de dicha ranura;
  - *Histograma de altura equilibrada*, que sitúa aproximadamente el mismo número de valores en cada ranura de modo que los puntos terminales de cada una de ellas se determinan según el número de valores contenidos en la ranura.

Optimización de consultas en Oracle: Histogramas



- (a) anchuras equilibradas. Cada ranura de igual anchura contiene dos valores (1-2, 3-4, etc.)
- (b) alturas equilibradas – la altura de cada columna es 20 (100/5)

Optimización de consultas en Oracle: Visualización del plan de ejecución

- La salida del optimizador es un plan de ejecución (secuencia de operaciones)
- Se puede visualizar con la sentencia EXPLAIN PLAN, que posee la siguiente información:
  - ordenación de las tablas referenciadas
  - método de acceso de cada tabla
  - método de unión de las tablas afectadas por operaciones de join
  - otras operaciones: filtros, clasificación, agregación.
- Los resultados se almacenan en la tabla PLAN_TABLE

Optimización de consultas en Oracle: Visualización del plan de ejecución

- La salida del optimizador es un plan de ejecución:

```
SQL> EXPLAIN PLAN
  2  SET STATEMENT_ID = 'PB'
  3  FOR SELECT b.branchNo, b.city, propertyNo
  4  FROM Branch b, PropertyForRent p
  5  WHERE b.branchNo = p.branchNo
  6  ORDER BY b.city;

Explained.

SQL> SELECT ID||' '||PARENT_ID||'   '||LPAD('   ', 2*(LEVEL – 1))||OPERATION||' '||OPTIONS||
  2  '   '||OBJECT_NAME "Query Plan"
  3  FROM Plan_Table
  4  START WITH ID = 0 AND STATEMENT_ID = 'PB'
  5  CONNECT BY PRIOR ID = PARENT_ID AND STATEMENT_ID = 'PB';

Query Plan
------------------------------------------------------------------
0     SELECT STATEMENT
1  0      SORT ORDER BY
2  1         NESTED LOOPS
3  2            TABLE ACCESS FULL PROPERTYFORRENT
4  2            TABLE ACCESS BY INDEX ROWID BRANCH
5  4               INDEX UNIQUE SCAN SYS_C007455

6  rows selected.
```

Bibliografía

- *Fundamentos de Sistemas de Bases de Datos (3º edic.). Ramez A. Elmasri, Shamkant B. Navathe*. [cap. 18]

- ***Fundamentos de Bases de Datos (5º edic.).*** **A. Silberschatz. McGraw-Hill [cap. 14]**

- *Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión*. T.Connolly, C. Begg, A. Strachan. Addison-Wesley