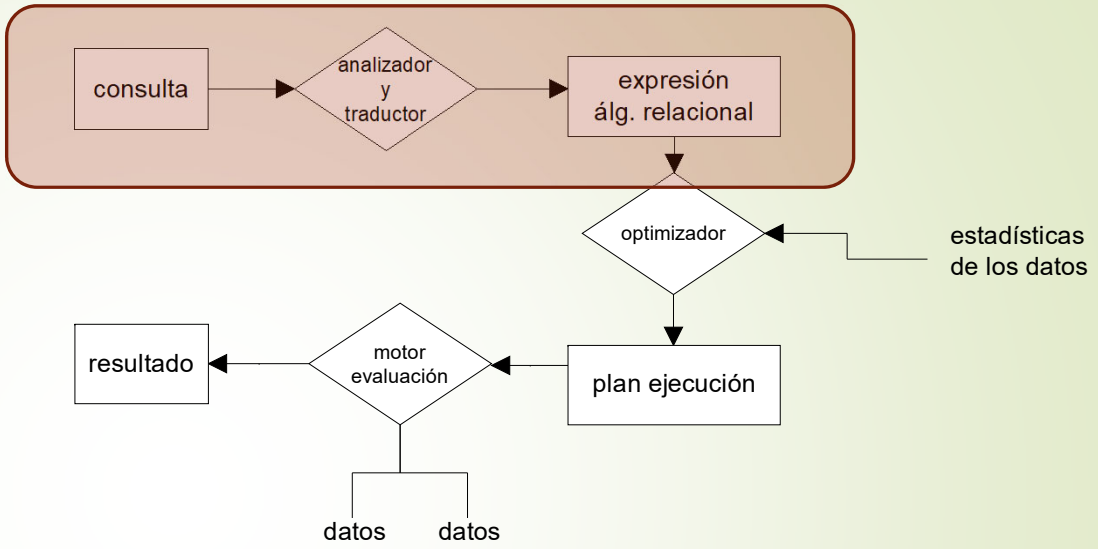
Tema.- Procesamiento y Optimización de consultas

Fundamentos de Bases de Datos (5º edic.).

A. Silberschatz. McGraw-Hill [caps. 13-14]

Procesamiento de consultas

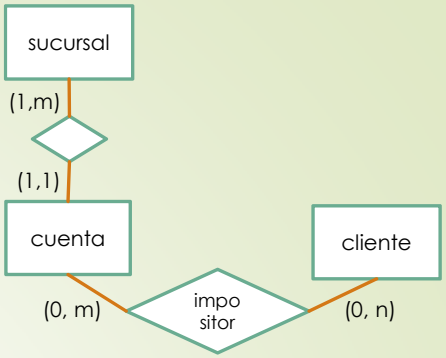
- Pasos:



Análisis y traducción:

- Comprobación de sintaxis, verificación de nombres
- Construcción del **árbol de consulta**:
  - Las **hojas** representan las **tablas** de la consulta
  - Los **nodos intermedios** son las **operaciones** de álgebra relacional. Cada operación da lugar a una tabla intermedia.
  - La **raíz** es la **respuesta** a la consulta.
- Transformación a una expresión en álgebra relacional

Procesamiento de consultas



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

Saldos menores de 2500

```
SELECT saldo
FROM cuenta
WHERE saldo < 2500
```

Análisis y traducción:

- Comprobación de sintaxis, verificación de nombres
- Construcción del **árbol de consulta**:
  - Las **hojas** representan las **tablas** de la consulta
  - Los **nodos intermedios** son las **operaciones** de álgebra relacional. Cada operación da lugar a una tabla intermedia.
  - La **raíz** es la **respuesta** a la consulta.

Π_{saldo}

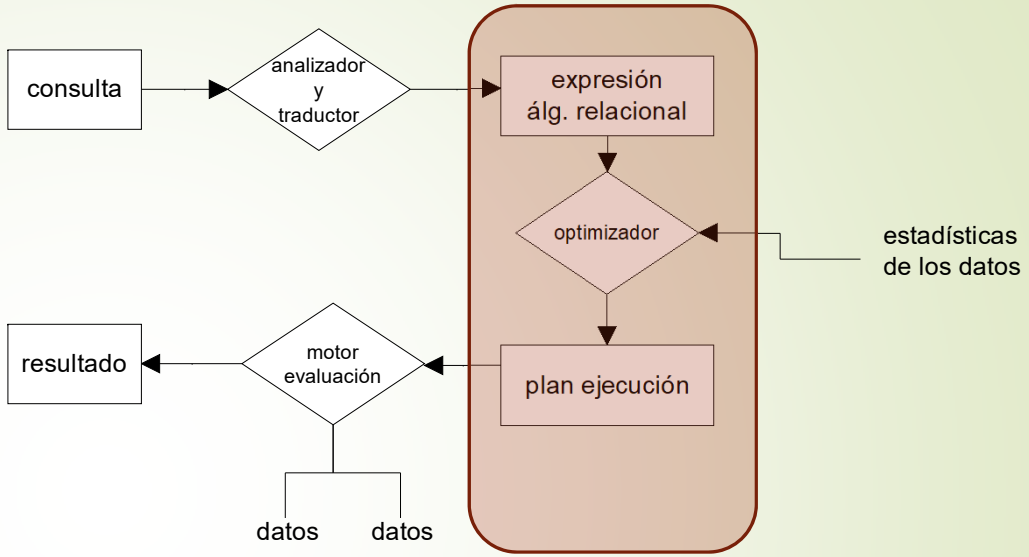
$\sigma_{saldo < 2500}$

cuenta

- Transformación a una expresión en álgebra relacional $\Pi_{saldo}(\sigma_{saldo < 2500}(cuenta))$

Procesamiento de consultas

- Pasos:



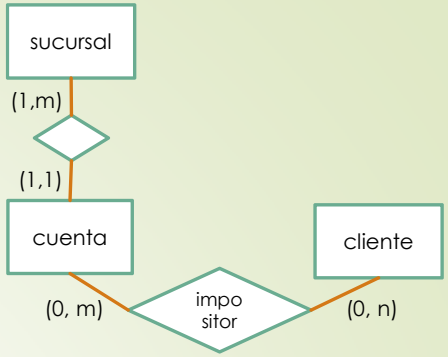
Análisis y traducción:

- Comprobación de sintaxis, verificación de nombres
- Construcción del árbol de consulta
- Transformación a una expresión en álgebra relacional

Optimización:

- Indicar cómo evaluar la expresión en álgebra relacional (algoritmo(s) a utilizar, índice(s) a aplicar,…) ⇒ **primitivas de evaluación** (operaciones álgebra anotadas)
- Determinar el **plan de ejecución/evaluación** (= secuencia de primitivas de evaluación) que minimice el coste de ejecución de la consulta

Procesamiento de consultas



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

Saldos menores de 2500

```
SELECT saldo
FROM cuenta
WHERE saldo < 2500
```

Optimización:

- Indicar cómo evaluar la expresión en álgebra relacional (algoritmo(s) a utilizar, índice(s) a aplicar,…) ⇒ **primitivas de evaluación** (operaciones álgebra anotadas)
- Determinar el **plan de ejecución/evaluación** (= secuencia de primitivas de evaluación) que minimice el coste de ejecución de la consulta

Π **saldo**

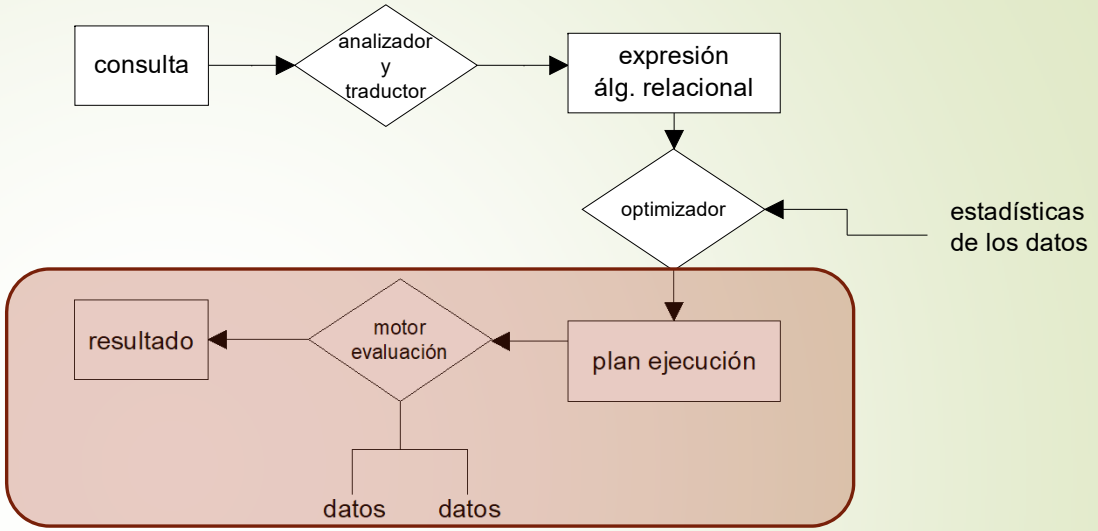
$\sigma_{\text{saldo} < 2500}$ **USAR ÍNDICE 1**

cuenta

2. **PLAN DE EJECUCIÓN/EVALUACIÓN**

Procesamiento de consultas

- Pasos:



Análisis y traducción:

- Comprobación de sintaxis, verificación de nombres
- Construcción del árbol de consulta
- Transformación a una expresión en álgebra relacional

Optimización:

- Indicar cómo evaluar la expresión en álgebra relacional (algoritmo(s) a utilizar, índice(s) a aplicar,…) ⇒ **primitivas de evaluación** (operaciones álgebra anotadas)
- Determinar el **plan de ejecución/evaluación** (= secuencia de primitivas de evaluación) que minimice el coste de ejecución de la consulta

Evaluación:

- El motor de consultas ejecuta un plan de ejecución / evaluación

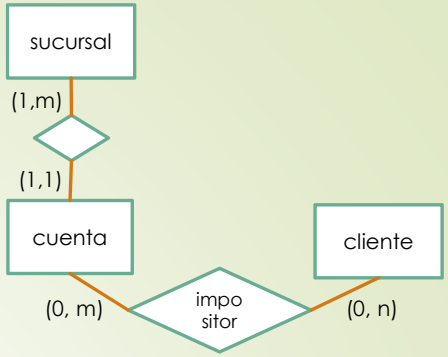
Optimización de consultas

Nombre de clientes con cuenta en una sucursal de Ourense

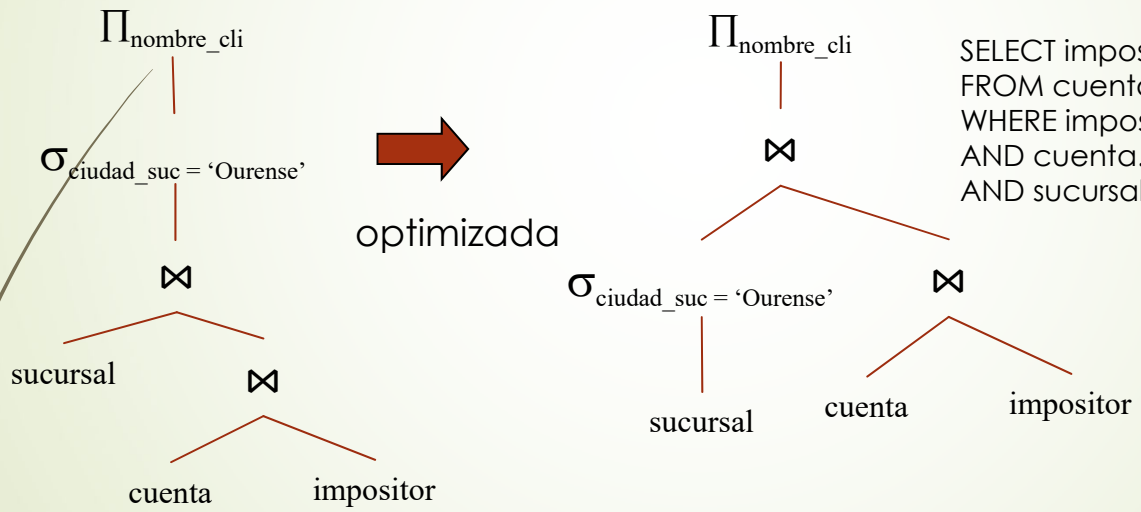
$\Pi_{nombre\_cli}\ (\sigma_{ciudad\_suc = 'Ourense'}\ (sucursal \bowtie (cuenta \bowtie impositor)))$

optimizada

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



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




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

El optimizador diseña un plan de evaluación "óptimo" generando planes alternativos

1. **Heurística**: genera expresiones equivalentes mediante ***reglas de equivalencia***
2. **Coste**: estima el coste de cada plan (utilizando inf. estadística de tablas, índices, etc.)

Transformación de expresiones relacionales

- Las consultas se pueden expresar de varias maneras, con costes de evaluación diferentes
  - En vez de tomar la expresión relacional original se consideran expresiones alternativas equivalentes

- 2 expresiones en álgebra relacional son **equivalentes** si generan el mismo conjunto de tuplas
  - Se puede sustituir la primera expresión por la segunda, o viceversa
  - Una expresión equivalente puede ser más eficiente que otra en tiempo de respuesta

- Para hallar expresiones equivalentes se usan **reglas o axiomas del álgebra relacional**
  - Provienen del álgebra de conjuntos

Reglas de equivalencia

1. **CASCADA DE σ:** las operaciones de selección conjuntivas pueden dividirse en una secuencia de selecciones individuales => $\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$

   ej: $\sigma_{\text{saldo>10} \wedge \text{nombre\_suc="Ourense"}}(\text{cuenta}) = \sigma_{\text{saldo>10}}(\sigma_{\text{nombre\_suc="Ourense"}}(\text{cuenta}))$

2. **CONMUTATIVIDAD DE σ:** las operaciones de selección son conmutativas => $\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$

   ej: $\sigma_{\text{saldo>10}}(\sigma_{\text{nombre\_suc="Ourense"}}(\text{cuenta})) = \sigma_{\text{nombre\_suc = "Ourense"}}(\sigma_{\text{saldo>10}}(\text{cuenta}))$

3. **CASCADA DE ∏:** solo es necesaria la última operación de proyección, las demás pueden omitirse => $\Pi_{L1}(\Pi_{L2}(\ldots\Pi_{Li}(E))\ldots)) = \Pi_{L1}(E)$

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

Reglas de equivalencia

4. **DISTRIBUTIVIDAD de Π y σ:**

$$\Pi_{L1}\ (\sigma_{\theta}(E_1)) = \sigma_{\theta}(\Pi_{L1}(E_1))\ \text{si}\ \theta \subseteq L1$$

ej: $\Pi_{\text{nombre\_suc, saldo}}(\sigma_{\text{saldo} > 10}(\text{cuenta})) = \sigma_{\text{saldo} > 10}(\Pi_{\text{nombre\_suc, saldo}}(\text{cuenta}))$

5. **CONMUTATIVIDAD de productos cartesianos y $\bowtie$:**

$$E1 \bowtie_{\theta} E2 = E2 \bowtie_{\theta} E1 \qquad E1 \times E2 = E2 \times E1$$

ej: **cuenta $\bowtie$ sucursal = sucursal $\bowtie$ cuenta**

ej: **cuenta X sucursal = sucursal X cuenta**

Reglas de equivalencia

6. **DISTRIBUTIVIDAD de σ con $\bowtie$ o con X**, bajo 2 condiciones:

   a) Si la selección θ solo implica atributos de una de las expresiones:

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

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

   b) Si la selección $\theta 1$ solo implica atributos de E1 y $\theta 2$ solo implica atributos de E2:

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

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

Esta regla permite bajar las **selecciones** hacia las hojas del árbol

Reglas de equivalencia

7. **DISTRIBUTIVIDAD de Π con $\bowtie$, bajo 2 condiciones:**

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

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

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

donde L1 son atributos de E1

L2 son atributos de E2

L3 son atributos de E1 que están en θ pero no en L1

L4 son atributos de E2 que están en θ pero no en L2

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

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

Esta regla permite bajar las **proyecciones** hacia las hojas del árbol

Reglas de equivalencia

8. **ASOCIATIVIDAD DE ⋈:** el join natural es asociativo => $(E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$

   ej: **(cuenta ⋈ sucursal) ⋈ impositor = cuenta ⋈ (sucursal ⋈ impositor)**

   Los zeta-join son asociativos en el siguiente sentido:

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

   ej: **(cuenta⋈ $_{c.num\_cta=i.num\_cta}$ impositor)⋈ $_{i.nombre\_cli=cl.nombre\_cli \wedge c.saldo > cl.sueldo}$ cliente=**

   = cuenta ⋈ $_{c.num\_cta = i.num\_cta \wedge c.saldo > cl.sueldo}$ (impositor ⋈ $_{i.nombre\_cli=cl.nombre\_cli}$ cliente)

9. **CONVERSIÓN del producto cartesiano en ⋈:**

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

   ej: $\sigma_{c.nombre\_suc = s.nombre\_suc}$ **(cuenta x sucursal) = cuenta ⋈** $_{c.nombre\_suc = s.nombre\_suc}$ **sucursal**

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

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

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:

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

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 Ourense es pequeño

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

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

Tipos de optimización: Heurística

- Mediante la aplicación de reglas de equivalencia, reordena los componentes del árbol de consultas inicial para intentar reducir el coste de la optimización
- Pasos a seguir:

1. **Desplazar las selecciones (σ) hacia las hojas para realizarlas tan pronto como sea posible**

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

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

   DISTRIBUTIVIDAD DE σ con $\bowtie$ ó X:

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

$\sigma_{\theta 0}(E1 \times E2) = \sigma_{\theta 0}(E1) \times E2$ 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 $\theta 2$ atributos de E2

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

Tipos de optimización: Heurística

2. Convertir el producto cartesiano (X) seguido de σ en $\bowtie$

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

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

3. Ejecutar cuanto antes las σ y $\bowtie$ que producen menos tuplas

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

ASOCIATIVIDAD DE $\bowtie$ $\Rightarrow (E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$

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

donde $\theta 2$ implica solo atributos de E2 y E3

4. Desplazar las proyecciones ($\prod$) hacia las hojas para realizarlas tan pronto como sea posible

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

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

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

Suele resultar mejor hacer las selecciones antes que las proyecciones porque:

- La selección tiene la posibilidad de reducir mucho el tamaño de las relaciones.
- **Solo se pueden aplicar índices sobre las tablas base**. Si se lleva a cabo la proyección antes se generaría una tabla intermedia y ya no se podrían utilizar índices

Ejemplo 1 uso reglas de equivalencia

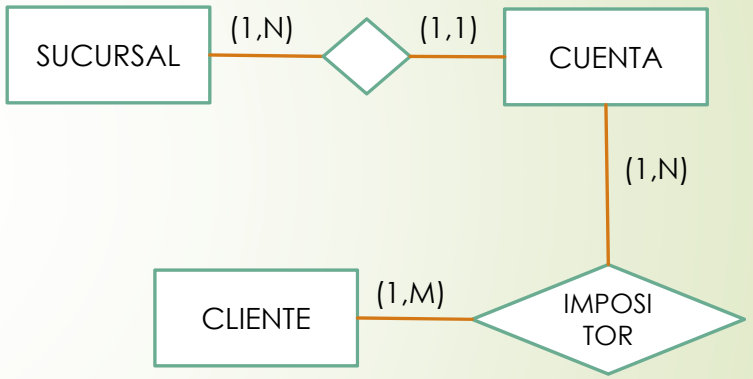
- Representar mediante árboles las expresiones del álgebra relacional de la consulta siguiente y transformarla a una forma más eficiente. Enunciar las reglas de equivalencia utilizadas en cada uno de los pasos del proceso

SUCURSAL (nombre_suc, ciudad_suc, activos)

CUENTA (num_cta, nombre_suc, saldo)

IMPOSITOR (nom_cli, num_cta)

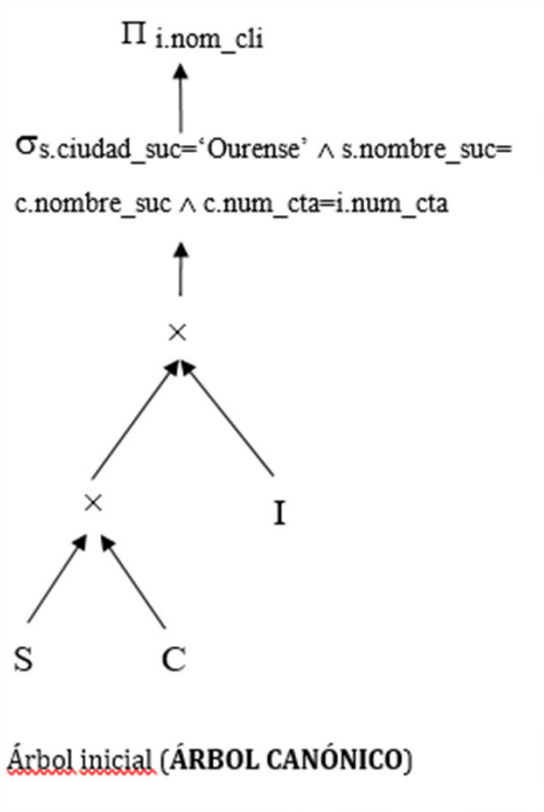
CLIENTE (nom_cli, direccion, activos)



Nombre de clientes con cuenta en alguna sucursal de Ourense

```
SELECT   nom_cli
FROM     SUCURSAL s, CUENTA c, IMPOSITOR i
WHERE    s.ciudad_suc = 'Ourense' AND
         s.nombre_suc = c.nombre_suc AND
         c.num_cta = i.num_cta;
```

Ejemplo 1 uso reglas de equivalencia



Árbol inicial (ÁRBOL CANÓNICO)

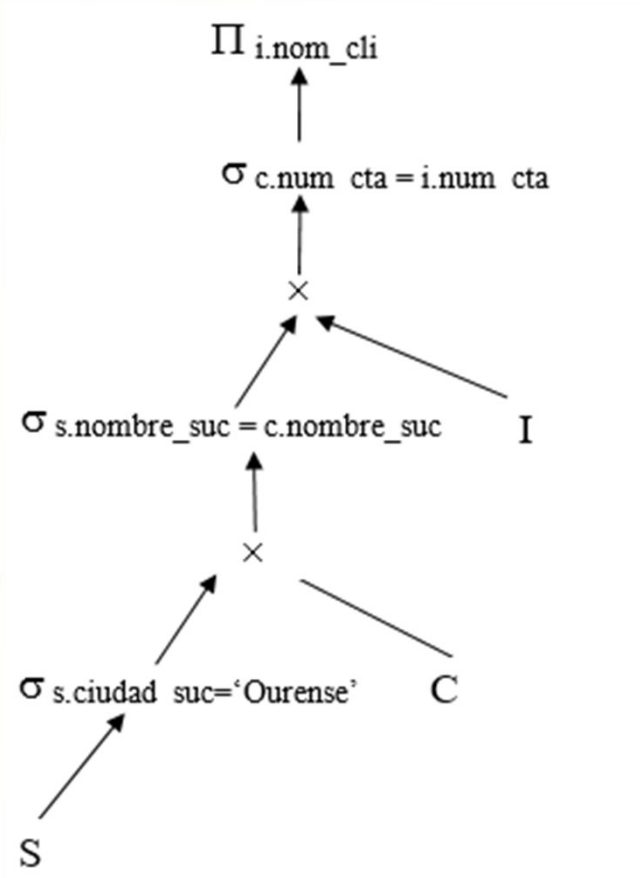
- **Árbol canónico**:
  - Árbol lineal izquierdo
    - En cada nodo binario el hijo derecho es una tabla
  - Cada nodo binario corresponde a un X
  - σ sobre toda la condición
  - Π sobre todos los atributos
- Es el más rápido de construir pero el más costoso

Expresión en álgebra relacional:

$\Pi_{\text{i.nom\_cli}} (\sigma_{\text{s.ciudad\_suc='Ourense' ^ s.nombre\_suc=c.nombre\_suc ^ c.num\_cta=i.num\_cta}}$

((sucursal X cuenta) X impositor))

Ejemplo 1 uso reglas de equivalencia



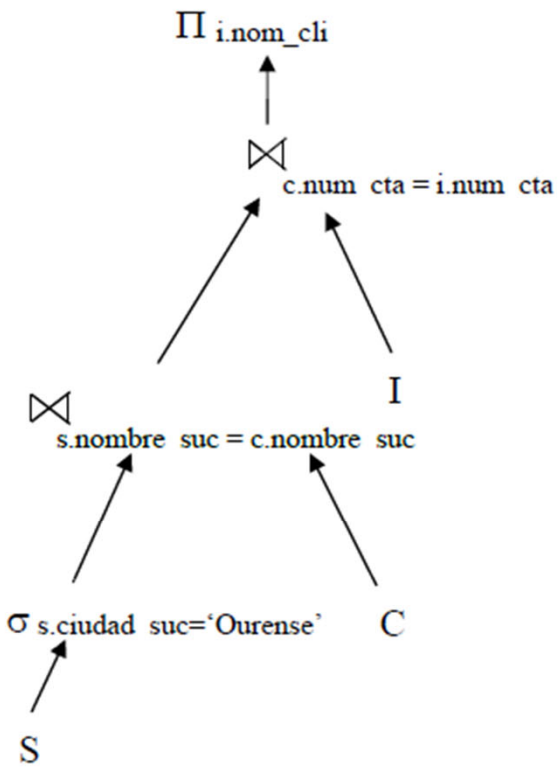
1. DESPLAZAR σ HACIA HOJAS EN X:
CASCADA DE σ: $\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$
CONMUTATIVIDAD DE σ: $\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$
DISTRIBUTIVIDAD DE σ CON X:
$\sigma_{\theta 0}(E1 X E2) = \sigma_{\theta 0}(E1) X E2$, θ0 implica solo atrib. de E1
$\sigma_{\theta 1 \wedge \theta 2}(E1 X E2) = \sigma_{\theta 1}(E1) X \sigma_{\theta 2}(E2)$ donde θ1 implica solo atrib. de E1 y θ2 atrib. de E2

Expresión en álgebra relacional

$\Pi_{i.nom\_cli}(\sigma_{c.num\_cta=i.num\_cta}(\sigma_{s.nombre\_suc=c.nombre\_suc}$

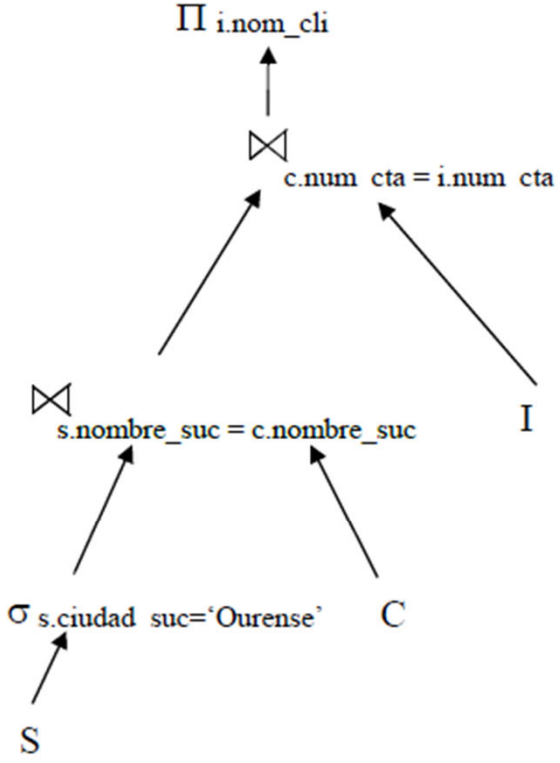
$(\sigma_{ciudad\_suc='Ourense'}(sucursal) X cuenta) X impositor))$

Ejemplo 1 uso reglas de equivalencia



2. Sustituir el producto cartesiano (X) seguido de σ por join:

$\sigma_\theta (E1\ X\ E2) = E1 \bowtie_\theta E2$



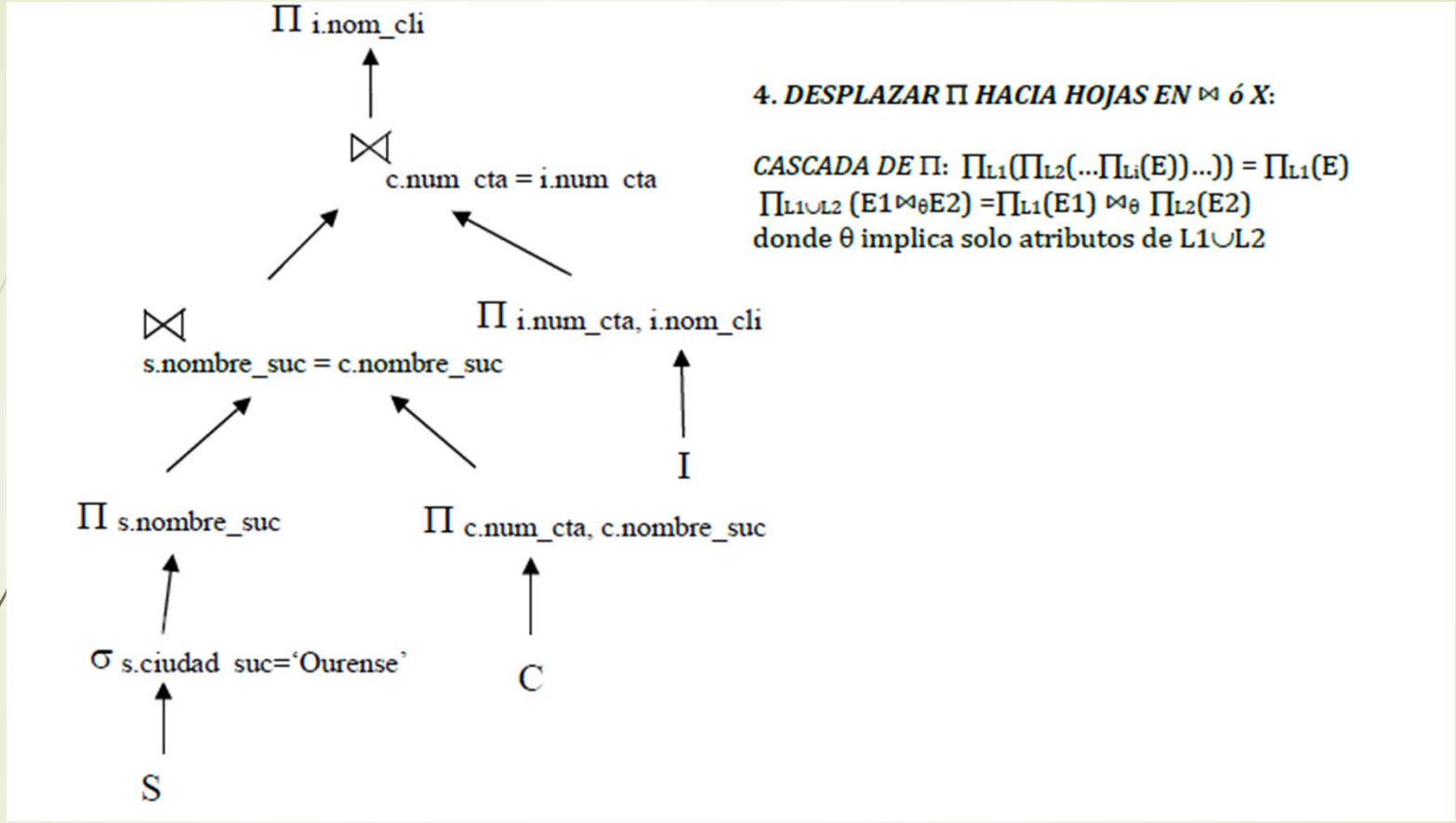
3. Ejecutar antes las selecciones y los joins que producen menos tuplas

CONMUTATIVIDAD DE σ => $\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$
ASOCIATIVIDAD DE ⋈ =>
$(E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$
En este caso, no es necesario aplicar las reglas

2. $\Pi_{nom\_cli}\ ((\sigma_{ciudad\_suc='Ourense'}\ (sucursal) \bowtie cuenta) \bowtie impositor)$

3. $\Pi_{nom\_cli}\ ((\sigma_{ciudad\_suc='Ourense'}\ (sucursal) \bowtie cuenta) \bowtie impositor)$

Ejemplo 1 uso reglas de equivalencia



4. *DESPLAZAR Π HACIA HOJAS EN ⋈ ó X*:

***CASCADA DE* Π:** $\prod_{L1}(\prod_{L2}(...\prod_{Li}(E))...)) = \prod_{L1}(E)$

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

donde θ implica solo atributos de L1∪L2

Expresión final optimizada

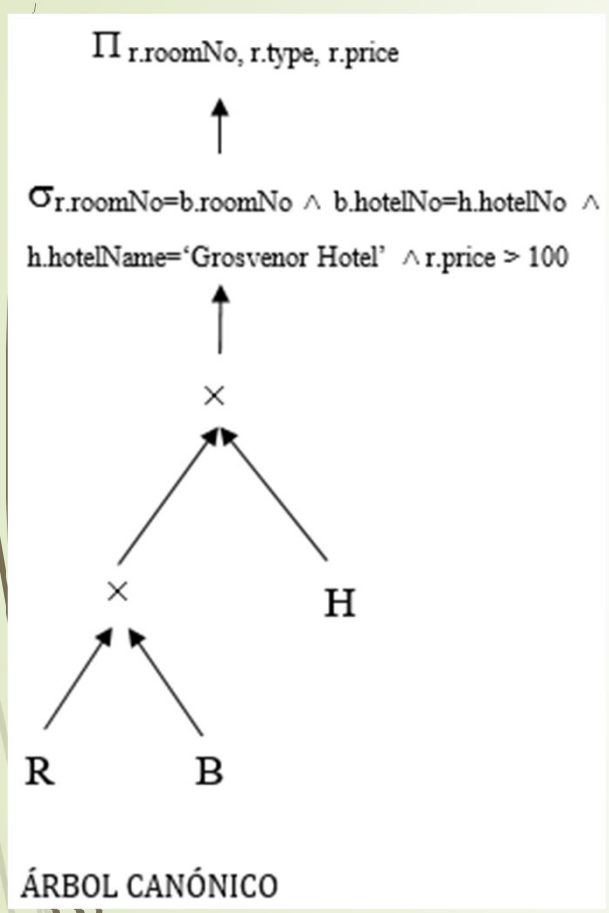
$$\prod_{nom\_cli}((\prod_{nombre\_suc}(\sigma_{ciudad\_suc='Ourense'}(sucursal)) \bowtie \prod_{num\_cta,nombre\_suc}(cuenta)) \bowtie \prod_{num\_cta,nom\_cli}(impositor))$$

Ejemplo 2 uso reglas de equivalencia

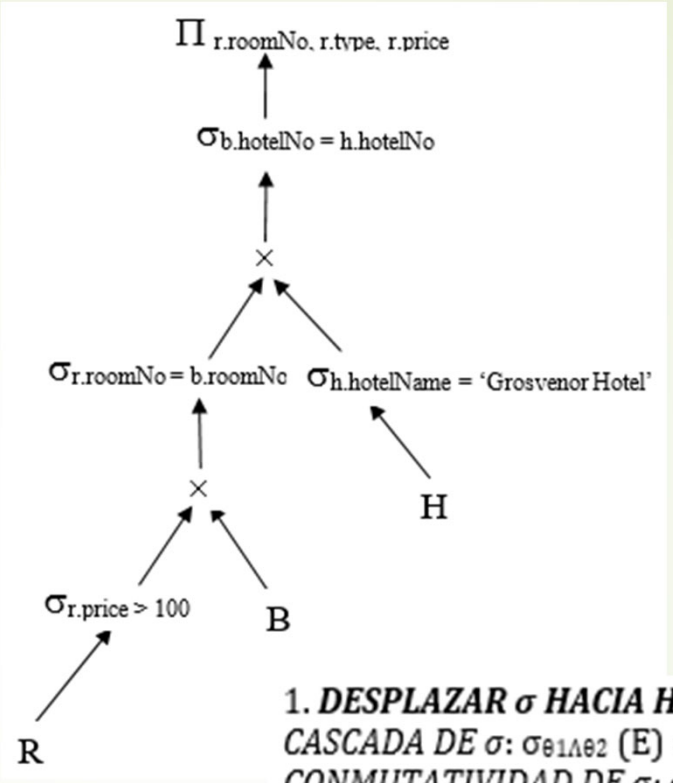
1 – Representar gráficamente mediante árboles las expresiones del álgebra relacional de la consulta siguiente, y transformarla a una forma más eficiente. Enunciar las reglas de equivalencia utilizadas en cada uno de los pasos del proceso:

```
SELECT r.roomNo, r.type, r.price
FROM Room r, Booking b, Hotel h
WHERE r.roomNo = b.roomNo AND b.hotelNo = h.hotelNo AND
      h.hotelName = "Grosvenor Hotel" AND r.price ≥ 100;
```

Ejemplo 2 uso reglas de equivalencia



ÁRBOL CANÓNICO



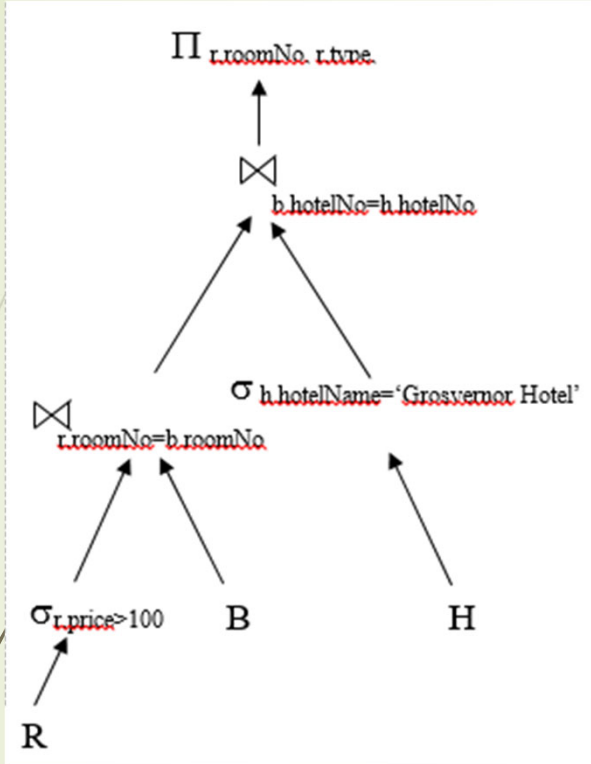
1. ***DESPLAZAR σ HACIA HOJAS EN X:***
CASCADA DE σ: $\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$
CONMUTATIVIDAD DE σ: $\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$
$\sigma_{\theta 0}(E1 X E2) = \sigma_{\theta 0}(E1) X E2$, θ0 implica solo atrib. de E1
$\sigma_{\theta 1 \wedge \theta 2}(E1 X E2) = \sigma_{\theta 1}(E1) X \sigma_{\theta 2}(E2)$ donde θ1 implica solo atributos de E1 y θ2 atributos de E2

Expresión álgebra relacional

$\Pi_{roomNo, type, price}(\sigma_{booking.hotelNo = hotel.hotelNo}(\sigma_{room.roomNo = booking.roomNo}$

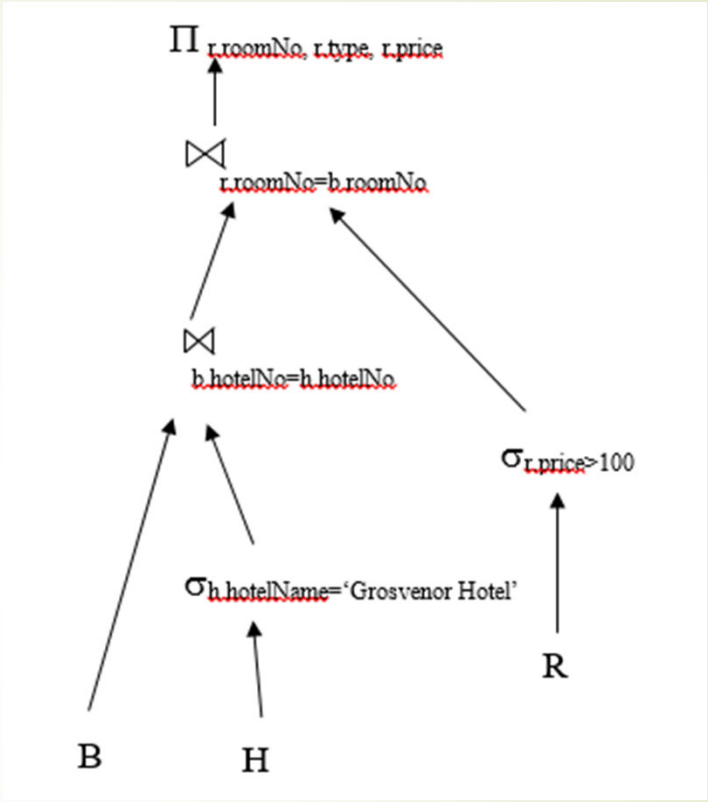
$(\sigma_{price >= 100}(Room) X Booking)) X (\sigma_{hotelName = `Grosvernor Hotel'}(Hotel))$

Ejemplo 2 uso reglas de equivalencia



2. Sustituir el producto cartesiano (X) seguido de σ por join:

$\sigma_\theta (E1 \; X \; E2) = E1 \bowtie_\theta E2$



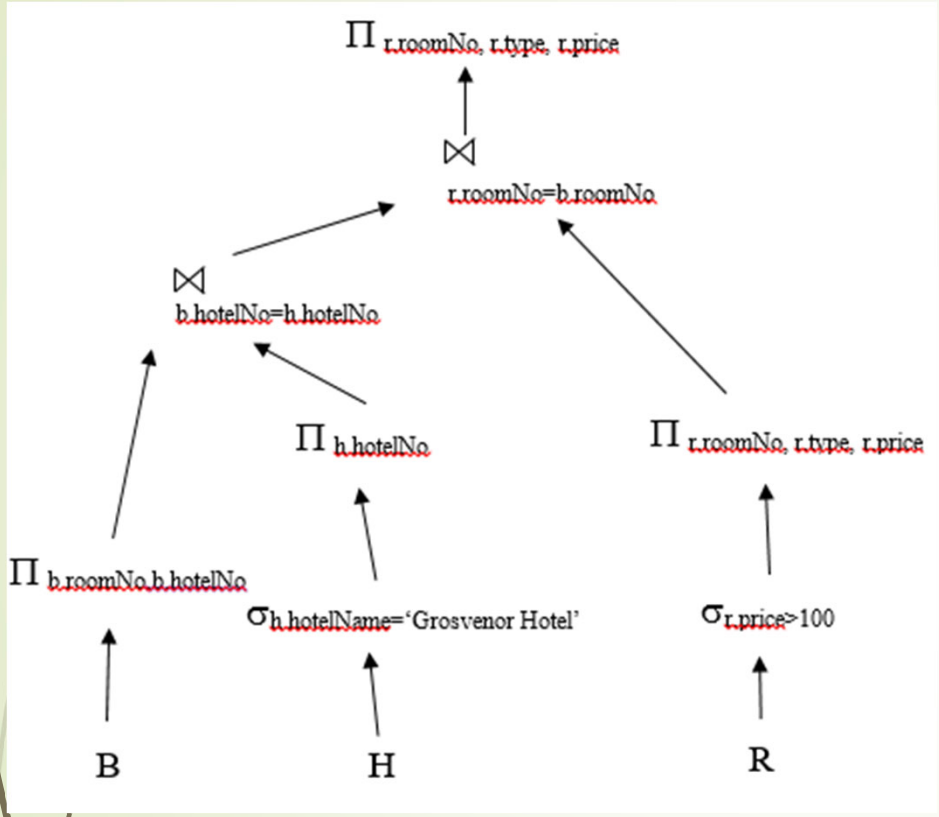
3. Ejecutar antes las selecciones y los joins que producen menos tuplas

(en este caso, el nº hoteles es inferior al nº habitaciones)

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

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

Ejemplo 2 uso reglas de equivalencia



4. *DESPLAZAR Π HACIA HOJAS EN ⋈ ó X*:

CASCADA DE Π: $\Pi_{L1}(\Pi_{L2}(...\Pi_{Li}(E))...)) = \Pi_{L1}(E)$

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

donde θ implica solo atributos de $L1 \cup L2$

Expresión final optimizada

$\Pi_{roomNo, type, price}$ (($\Pi_{roomNo, hotelNo}$ (Booking) ⋈ $\Pi_{hotelNo}$ ($\sigma_{hotelName = `Grosvernor Hotel'}$ (Hotel))) ⋈ $\Pi_{roomNo, type, price}$ ($\sigma_{price >= 100}$ (Room)))