Unidad V. Diseño de Bases de Datos Distribuidas

M. en C. Euler Hernández Contreras

Contenido

1. Alternativas de estrategias de diseño
   a) Top-Down
   b) Botton-Up

2. Fragmentación
   a) Razones para fragmentar
   b) Alternativas de fragmentación
   c) Reglas de Fragmentación

3. Tipos de Fragmentación
   a) Fragmentación Horizontal
   b) Fragmentación Vertical
   c) Fragmentación Híbrida

4. Localización
   a) Problema de localización
   b) Requerimientos de Información
   c) Modelo de localización

Referencia Bibliográfica

1. M. Tamer Özsu, Patrick Valduriez. Principles of Distributed Database Systems. Second Edition. Prentice-Hall, Estados Unidos. 1999, págs. 666

2. Stefano Ceri, Giuseppe Pelagatti. Distributed Databases Principles & Sistems. Mc Graw-Hill Inc., 1985, págs. 385

Alternativas de Estrategias de Diseño[1][2]

Dos estrategias han sido identificadas para el diseño de bases de datos distribuidas integración lógica por medio de diseño *Top-Down* y *Bottom-Up*.

a) Integración lógica por medio de diseño Top – Down

No existe un diseño como tal ya que se inicia desde cero; el diseñador requiere identificar las tablas, pero también su ubicación y la necesidad de considerar replicación (Ver Figura 1) tal como se explicó en el Modelo de Referencia para un Sistema de Bases de datos Distribuidas en la unidad anterior.

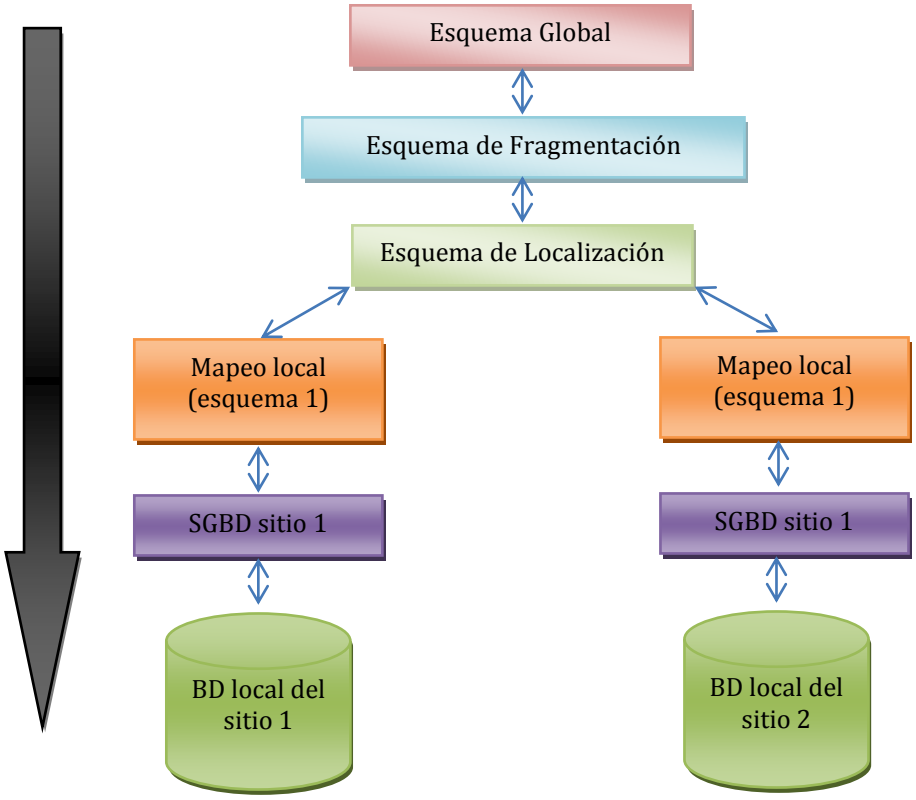


Figura 1. Integración Lógica por medio de diseño *Top-Down*

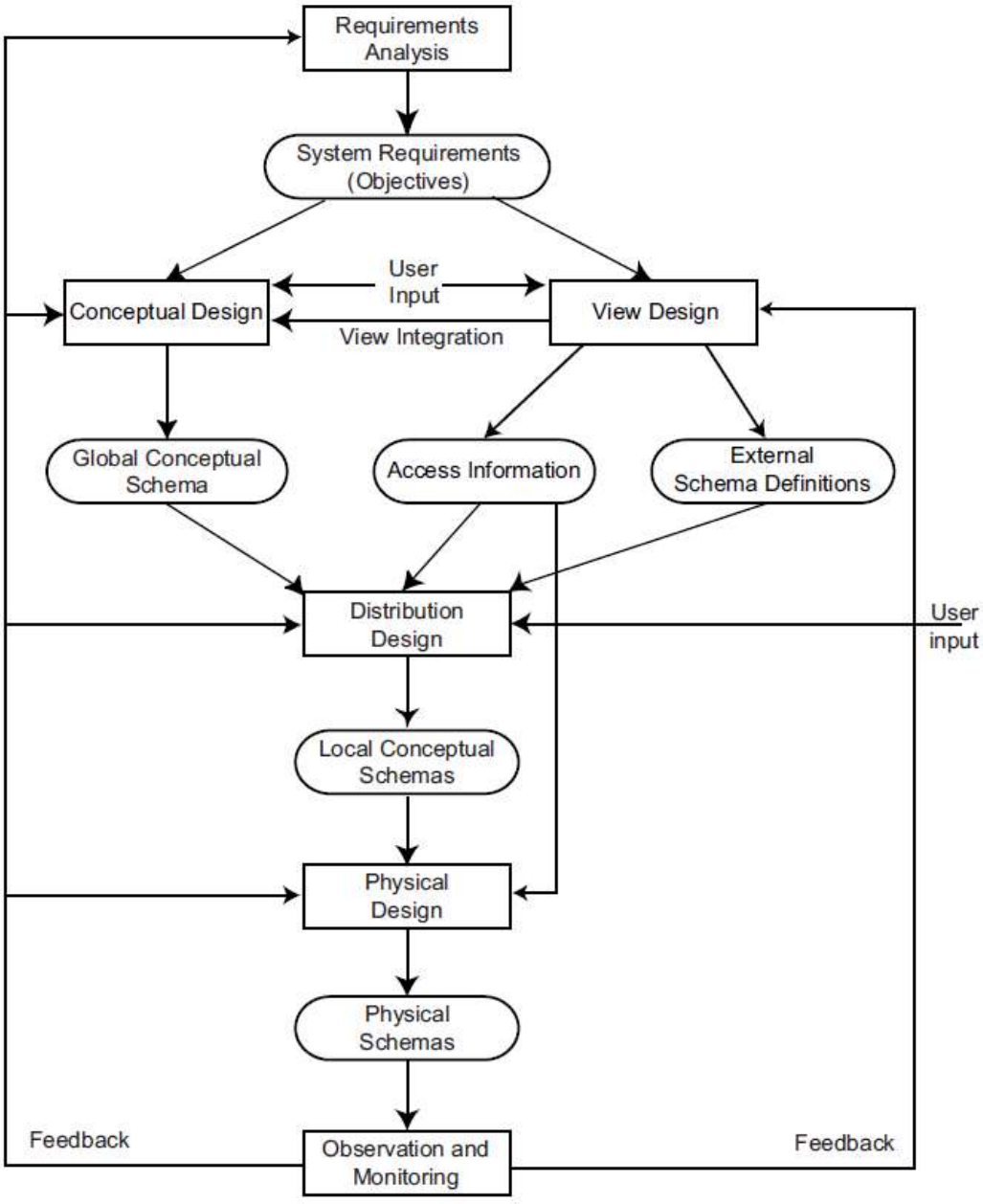


Figura 2. *Top-Down Design Process*

b) Integración lógica por medio de diseño Bottom – up

Se construye la BDD partiendo de BD existentes (multibase); se requiere de integración de esquemas existentes de diversos sitios para construir el esquema global. Este tipo de ambiente existe principalmente en el contexto de bases de datos heterogéneas (Ver Figura 3).

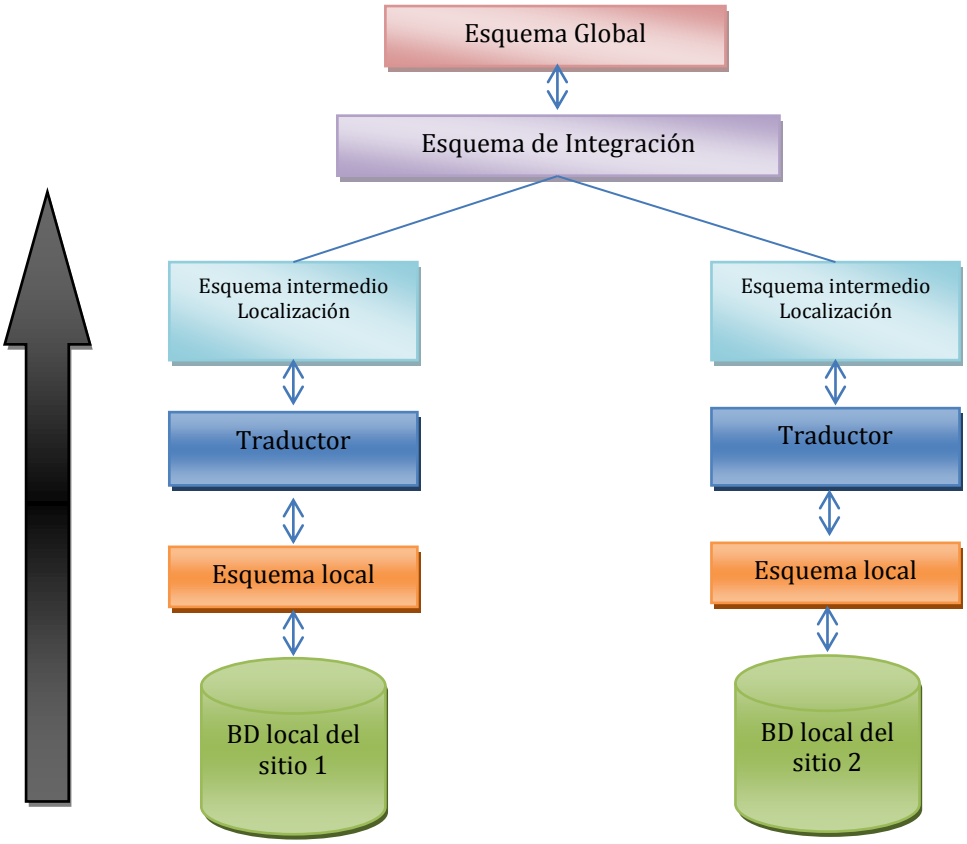


Figura 3. Integración Lógica por medio de diseño *Bottom-Up*

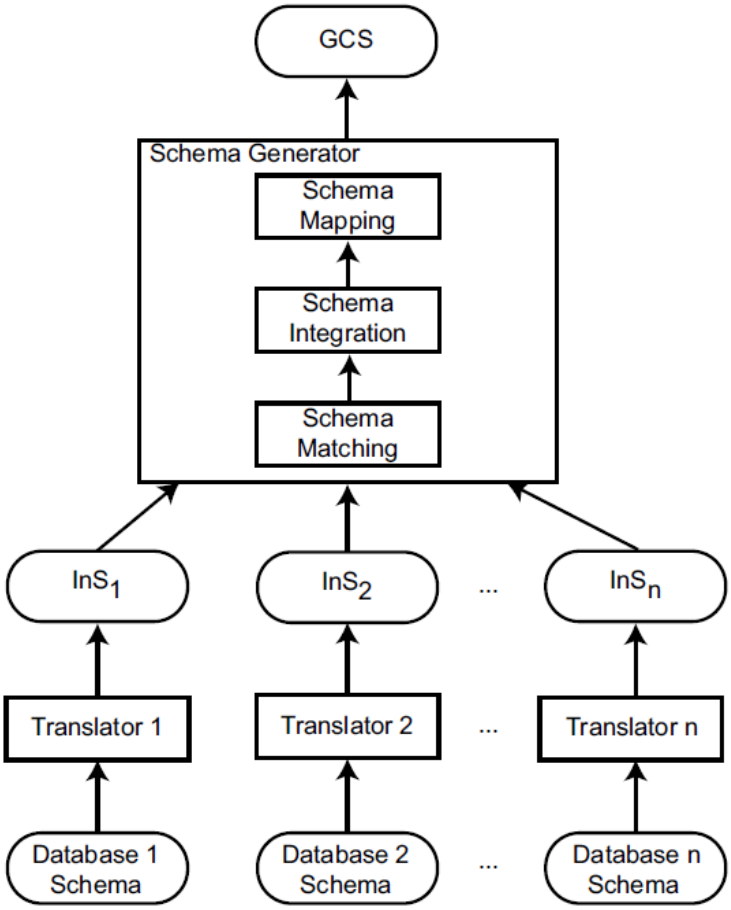


Figura 3. *Database Integration Process* (*Bottom-Up Design Process)*

El diseño *Botton-up* involucra el proceso en el que la información que reside en diversas bases de datos (física o lógica) pueden ser integradas para formar una simple multibase de datos.

Fragmentación[1]

Razones para llevar a cabo Fragmentación

Para llevar a cabo el proceso de Fragmentación es importante responder las siguientes preguntas.

1. ¿Por qué Fragmentar?
2. ¿Cómo Fragmentar?
3. Fragmentar, ¿a qué nivel?
4. ¿Qué tan correcta o apropiada es la Fragmentación?
5. ¿Cómo asignar fragmentos en cada sitio?
6. ¿Qué información es necesaria para fragmentar y localizar los fragmentos en un sitio determinado?

Alternativas de Fragmentación

Existen claramente dos alternativas para llevar a cabo la división de tablas a otras más pequeñas: Fragmentación horizontal y vertical.

Reglas de Fragmentación

a) Completitud

Si la instancia de una relación R es descompuesta en fragmentos $R_1, R_2, \ldots, R_n$, cada elemento de dato que puede ser encontrado en R, puede ser encontrado en una o más R_i.

Características:

- Esta propiedad es idéntica a la pérdida de descomposición en el proceso de normalización
- Asegura que los datos de una relación global es mapeado en fragmentos sin pérdida
- El término “elemento” en la fragmentación Vertical hace referencia a atributo, mientras que la fragmentación Horizontal a tupla.

b) Reconstrucción

Si en una relación R es descompuesto en fragmentos $R_1, R_2, \ldots, R_n$, es posible definir un operador relacional ∇ tal que:

$$R = \nabla R_i, \ \forall R_i \in F_R$$

El operador ∇será diferente para las diferentes formas de fragmentación. La reconstrucción asegura que el contenido definido en los datos en la forma de dependencia son preservados.

c) Fragmentos disjuntos

Si en una relación R es horizontalmente descompuesto en fragmentos $R_1, R_2, \dots, R_n$, y el elemento de datos d_i, está en R_j, éste no se encuentra en cualquier otro fragmento R_k $(k \neq j)$.

Este criterio asegura que los fragmentos horizontales son disjuntos. Si la relación es verticalmente descompuesta, los atributos que forman la llave primaria aparecerán en todos los fragmentos, en este caso, la disyunción en la fragmentación vertical se define en los atributos no primos.

Tipos de Fragmentación

a) Fragmentación Horizontal
b) Fragmentación Vertical
c) Fragmentación Híbrida (también conocida como Mixta)

Fragmentación Horizontal

Como se explico previamente, cada fragmento contiene un conjunto de tuplas de una relación. Existen dos versiones de Fragmentación horizontal:

Primaria:
Es llevada a cabo usando predicados definidos en una relación

Derivada:
Es la división (partición) de una relación que resulta de aquellos predicados definidos de otra relación.

¿Qué información es requerida para llevar a cabo Fragmentación horizontal?

a) Conocer el esquema relacional
b) Considerar los enlaces entre los objetos de la BD

Se definen dos funciones *Miembro* y *Propietario*, estos proporcionan información acerca del enlace entre un conjunto de relaciones (Ver Figura 4).

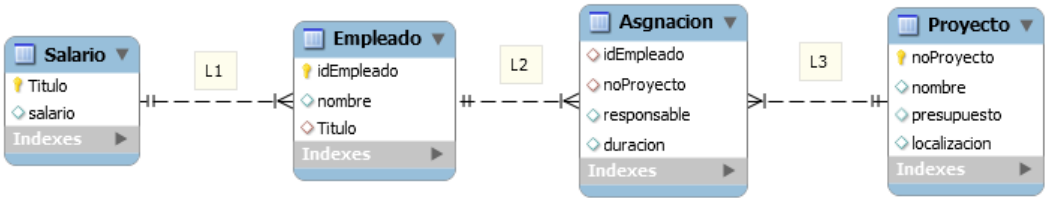


Figura 4. Modelo Relacional donde se identifican Funciones Miembro y Propietario

La relación Miembro se considera aquella relación que contiene la FK que hace referencia a una relación base, de lo contrario la relación Propietario es aquella relación base donde una relación miembro lo referencia.

Ejemplos:
Para el enlace L_1 tenemos que:
Propietario (L_1) = Salario (Pk)
Miembro (L_1) = Empleado (Fk)

Para el enlace L_2 tenemos que:
Propietario (L_2) = Empleado (Pk)
Miembro (L_2) = Asignación (Fk)

Para el enlace L_3 tenemos que:
Propietario (L_3) = Proyecto (Pk)
Miembro (L_3) = Asignación (Fk)

c) La información requerida acerca de la BD es la cardinalidad de cada relación *R*, denotado como *card(R)*

Luego entonces la información requerida a partir de una aplicación es la siguiente:

a) Información Cuantitativa: nos lleva a definir el modelo de localización.
b) Información Cualitativa: nos lleva a la actividad de la fragmentación, donde se hace uso de predicados en las consultas.

Considerando las instancias del modelo relacional (Figura 4), tenemos los siguiente tuplas.

Empleado

noEmp	nombre	Titulo
E1	P.P.	Ing Eléctrico
E2	K.L.	Ing en Sistemas
E3	A.L.	Ing Mecánico
E4	J.M.	Programador
E5	B.C.	Ing en Sistemas
E6	L.C	Ing Eléctrico
E7	R.D.	Ing Mecánico
E8	E.H.	Ing en Sistemas

Asignación

noEmp	noProyecto	Responsable	Duración
E1	**P1**	Administrador	12
E2	**P1**	Analista	24
E2	**P2**	Analista	6
E3	**P3**	Consultor	10
E3	**P4**	Ingeniero	48
E4	**P2**	Administrador	18
E5	**P2**	Administrador	24

Proyecto.

noProyecto	Nombre	Presupuesto	localización
P1	Instrumentación	150,000	México
P2	Desarrollo de BD	135,000	Monterrey
P3	Diseño asistido por computadora.	250,000	Monterrey
P4	Mantenimiento.	310,000	Puebla

Salario.

Título	Salario
Ing. Eléctrico	40,000
Ing. En Sistemas.	34,000
Ing. Mecánico.	27,000
Programador.	24,000

Un punto importante es obtener:

1. Predicados Simples:

Dada una relación $R(A_1, A_2, \ldots, A_n)$, donde: A_i es un atributo definido sobre el dominio D_i, un ***predicado simple*** p_j definido en R tiene la forma

$$p_j: A_i\ \theta\ Valor$$

Donde $\theta \in \{ =, <, \neq, \leq, >, \geq \}$ y $Valor \in Di$. Para la relación R se define un conjunto de predicados simples como $Pr = \{ p_1, p_2, \ldots, p_m \}$.

Las siguientes expresiones se consideran como predicados simples a partir de la relación Proyecto.

p_1: Nombre = "Mantenimiento"
p_2: Presupuesto < 200,000

2: Obtener Predicados Minitérminos:

Dado la relación R y el conjunto de predicados simples $Pr_i = \{ p_{i1}, p_{i2}, \ldots, p_{im} \}$, se define el conjunto de ***predicados minitérmino*** como $M = \{ m_{i1}, m_{i2}, \ldots, m_{iz} \}$ como

$$M_i = \{ m_{ij} \mid m_{ij} = \bigwedge p_{ik} \in Pr_i \ \ p_{ik}^* \}, 1 \leq k \leq m, 1 \leq j \leq z$$

Donde $p_{ik}^* = p_{ik}$ o $p_{ik}^* = \neg\, p_{ik}$. Así que cada predicado simple puede ocurrir en un predicado minitérmino en su *forma natural* o en su *forma negada*.

Es muy importante notar un punto aquí. La referencia de la negación de un predicado es significante para la forma natural *atributo=valor*. La negación tiene que ser tratado como el complemento. Por ejemplo la negación de este simple predicado *atributo≤valor* sería *atributo>valor*.

Ejemplo: Los siguientes son minitérminos de la relación Proyección descrita previamente.

m_1: Nombre = "Mantenimiento" $\wedge$ Presupuesto ≤ 200,000
m_2: $\neg$ (Nombre = "Mantenimiento") $\wedge$Presupuesto ≤ 200,000
m_3: Nombre = "Mantenimiento" $\wedge \neg$ (Presupuesto ≤ 200,000)
m_4: $\neg$ (Nombre= "Mantenimiento") $\wedge \neg$ (Presupuesto ≤ 200,000)

a) Del ejemplo anterior no son todos los minitérminos que se pueden definir, solo son algunos de ellos

b) Algunos de ellos no muestran congruencia (o carecen de significado semántico) de la relación Proyecto

Nota: m_2 puede reescribirse de la siguiente manera
m_2: Nombre != "Mantenimiento" ^ presupuesto ≤ 200,000

En términos de la información cuantitativa acerca de las aplicaciones de usuario, se necesita tener dos conjuntos de datos:

1. La *selectividad de los minitérminos*: Denotada como *sel*(m_i), se refiere al número de tuplas de la relación que serán accedidas por una consulta de usuario definida de acuerdo a un predicado minitérmino m_i.

2. La *frecuencia de acceso*: Si $Q=\{q_1, q_2,..., q_n\}$ es un conjunto de consultas entonces *acc*(q_i) indica la frecuencia con la cual una consulta de usuario q_i es accedida en un periodo de tiempo. Note que las frecuencias de acceso de minitérminos se pueden determinar a partir de las frecuencias de consultas. La frecuencia de acceso de un minitérmino se denota como *acc*(m_i).

Fragmentación Horizontal Primaria[1]

Una fragmentación horizontal primaria se define por una operación de selección en las relaciones propietarias de un esquema de la base de datos. Por tanto, dada una relación *R*, su fragmentación horizontal está dada por:

$$R_i = \sigma F_i (R), 1 \leq i \leq w$$

Donde, F_i es una fórmula de selección, la cual es preferiblemente un predicado minitérmino. Por lo tanto, un fragmento horizontal R_i de una relación *R* consiste de todas los tuplas de *R* que satisfacen un predicado minitérmino m_i. Lo anterior implica que dado un conjunto de predicados minitérmino *M*, existen tantos fragmentos horizontales de *R* como minitérminos existan. El conjunto de fragmentos horizontales también se entiende como los ***fragmentos minitérminos***.

Es necesario desarrollar un algoritmo que tome como entrada una relación *R* y el conjunto de predicados simples *Pr* y proporcione como resultado el conjunto de fragmentos de $R = \{ R_1, R_2, ..., R_m \}$ el cual obedece las reglas de fragmentación. Un aspecto importante del conjunto de predicados es que debe ser ***completo*** y ***mínimo***.

a) Completitud

Un conjunto de predicados simples *Pr* se dice que es ***completo*** si y solo si los accesos a los tuplas de los fragmentos minitérminos definidos en *Pr* requieren que dos tuplas del mismo fragmento tengan la misma probabilidad de ser accedidos por cualquier aplicación.

Considerando la relación Proyecto, se pueden definir consultas para (1) encontrar los presupuestos de los proyectos en base a la localización y (2) encontrar proyectos con presupuestos menores a $20,000.

Considerando (1) tenemos que:

P_r = {localización = "México", localización = "Monterrey", localización = "Puebla"}

No es completa con respecto a (2) ya que algunas tuplas de Proyecto tienen una probabilidad mayor de ser accedidos por la segunda consulta.

Si se modifica *Pr* como:

P_r = {localización = "México", localización = "Monterrey", localización = "Puebla", presupuesto <= 20,000, presupuesto > 20000}

… entonces P_r es completo.

b) Minimalidad

De manera intuitiva se puede ver que si un predicado influye en la fragmentación, esto es, causa que un fragmento *f*, se descomponga en f_1, y f_2, entonces habría una consulta que accede f_i y f_j de diferente manera.

En otras palabras, un predicado debe ser relevante en determinar una fragmentación. Si todos los predicados de un conjunto *Pr* son *relevantes*, entonces *Pr* es ***mínimo***.

Relevancia:

- Sean m_i y m_j dos predicados minitérminos definidos exactamente igual excepto que m_i contiene a p_i (forma natural) y m_j contiene a $\neg p_j$, también sean f_i y f_j los dos fragmentos definidos de acuerdo a m_i y m_j respectivamente. Entonces p_i es relevante si y solo si:

$$\frac{acc\ (m_i)}{card\ (f_i)} \neq \frac{acc\ (m_j)}{card\ (f_j)}$$

Por ejemplo, el conjunto P_r definido es mínimo y completo; sin embargo si se le agrega el predicado nombre = "Instrumentación" entonces, P_r no es mínimo ya que no hay una aplicación (consulta) que accedería a los fragmentos resultantes.

El algoritmo siguiente llamado COM_MIN genera un conjunto completo y mínimo de predicados *Pr'* dado un conjunto de predicados simple *Pr*. Por brevedad durante el algoritmo se utiliza la siguiente regla:

Regla 1: *Regla fundamental de completitud y minimalidad*, la cual afirma que una relación o fragmento es particionado en al menos dos partes las cuales se acceden en forma diferente por al menos una consulta de usuario.

Algoritmo COM_MIN

Entrada:
Una relación *R* y un conjunto de predicados simples *Pr.*

Salida:
Un conjunto completo y mínimo de predicados simples *Pr'* para *Pr*.

Declarar:
F: un conjunto de Fragmentos Minitérminos.

begin
Encontrar un $p_i \in Pr$ tal que p_i particiona a *R* de acuerdo a la regla 1.
$Pr' = pi$;
$Pr = Pr - pi$;
$F = f_i$ { f_i es un fragmento minitérmino de acuerdo a p_i }
do {Iterativamente agregar predicados a *Pr'* hasta que sea completo}
Encontrar un $p_j \in Pr$ tal que p_j particiona algún f_k de *Pr'* de acuerdo a la regla 1.
$Pr' = Pr' \cup p_j$;
$Pr = Pr - p_j$;
$F = F \cup f_i$;
if $\exists\ p_k \in Pr'$ el cual es no relevante *then*,
begin
$Pr' = Pr' - p_k$;
$F = F - f_k$
end-if
end-begin
until Pr' es completo
end {*COM_MIN*}

El algoritmo empieza encontrando un predicado que es relevante y particiona la relación de entrada. Después, agrega de manera iterativa predicados a este conjunto, asegurando minimalidad en cada paso. Por lo tanto, al final el conjunto *Pr'* es tanto completo como mínimo.

El segundo paso en el proceso de diseño de fragmentación horizontal primaria es derivar el conjunto de predicados minitérminos que pueden ser definidos en los predicados del conjunto *Pr'*. Esos minitérminos definen los fragmentos que serán usados como candidatos en el paso de asignación (localización).

Obtener predicados minitérminos es trivial, la dificultad radica en el número posibles de predicados que se puede generar, por lo que es necesario reducir el número de predicados minitérminos que tienen que ser considerados en la fragmentación.

Esta reducción puede ser alcanzada al eliminar fragmentos minitérminos que pueden carecer de significado. La eliminación se realiza al identificar minitérminos que pueden ser contradictorios con base a un conjunto de implicaciones *I*. Por ejemplo, si $Pr'=\{p_1,p_2\}$, entonces

p_1: *atributo=valor_1*
p_2: *atributo=valor_2*

Y el dominio del atributo es {*valor_1, valor_2*}, es obvio que *I* contiene dos implicaciones:

i_1: (atributo=valor_1) $\Rightarrow \neg$ (atributo = valor_2)
i_2: $\neg$ (atributo=valor_1) $\Rightarrow$ (atributo = valor_2)

Los siguientes cuatro predicados minitérminos pueden definirse de acuerdo a *Pr'*:

m_1: (atributo=valor_1) $\wedge$ (atributo = valor_2)
m_2: (atributo=valor_1) $\wedge \neg$ (atributo = valor_2)
m_3: $\neg$ (atributo=valor_1) $\wedge$ (atributo = valor_2)
m_4: $\neg$ (atributo=valor_1) $\wedge \neg$ (atributo = valor_2)

En este caso los predicados minitérminos m_1 y m_4 son contradictorios a las implicaciones I y pueden ser eliminados del conjunto *M*.

El algoritmo de fragmentación horizontal primaria, llamado **PHORIZONTAL**, se presenta a continuación. La entrada al algoritmo es una relación R_i la cual es sometida a fragmentación horizontal primaria, y Pr_i, el cual es el conjunto de predicados simples que han sido determinados de acuerdo a las consultas definidas en la relación R_i.

Algoritmo PHORIZONTAL

Entrada:
Una relación *R* y un conjunto de predicados simples *Pr*.

Salida:
Un conjunto de predicados minitérminos *M*.

begin

1. *Pr'* =COM_MIN(*R*, *Pr*)
2. Determinar el conjunto *M* de predicados minitérminos

3. Determinar el conjunto *I* de implicaciones entre $p_i \in Pr'$
for each $m_i \in M$ *do*
if m_i, es contradictorio de acuerdo a *I then*
{eliminar minitérminos contradictorios a partir de *M*}
$M = M - m_i$;
end-if
end-for
end {PHORIZONTAL}

Ejemplo 1:

Para la relación Salario la consulta o aplicación es verificar la información del salario y determinar incrementos. Suponga además que los registros de empleados se mantienen en dos lugares y, por tanto, la aplicación o consulta se ejecuta en dos lugares.

Los predicados simples que serían usados para particionar la relación Salario son:

$$p_1 : \text{SAL} \leq 30000$$
$$p_2 : \text{SAL} > 30000$$

Al aplicar el algoritmo COM_MIN se verifica que $Pr = \{ p_1, p_2 \}$ es completo y mínimo, $Pr' = Pr$.

Se pueden formar los siguientes predicados minitérminos como miembros de *M*:

m_1: (Salario ≤ 30000) ∧ (Salario > 30000)
m_2: (Salario ≤ 30000) ∧ ¬ (Salario > 30000)
m_3: ¬ (Salario ≤ 30000) ∧ (Salario > 30000)
m_4: ¬ (Salario ≤ 30000) ∧ ¬ (Salario > 30000)

Asumiendo que el dominio de SALARIO se puede partir en dos, como se sugiere *Pr* p_1 y p_2, las siguientes implicaciones son obvias:

i_1: (Salario ≤ 30000) ⇒ ¬ (Salario > 30000)
i_2: ¬ (Salario ≤ 30000) ⇒ (Salario > 30000)
i_3: (Salario > 30000) ⇒ ¬ (Salario ≤ 30000)
i_4: ¬ (Salario > 30000) ⇒ (Salario ≤ 30000)
De acuerdo a i_1, m_1 es contradictorio; de acuerdo a i_2, m_4 es contradictorio. Por lo tanto, nos quedamos con $M = \{ m_2, m_3 \}$. Por tanto, se definen los dos fragmentos *Fsalario* = { $Salario_1$, $Salario_2$ } de acuerdo a *M*.

Ejemplo 2:

Para la relación Proyecto la consulta es encontrar el nombre y presupuesto de proyectos dados por su número. Esta consulta es realizada en tres lugares. El acceso a la información de proyecto se realiza de acuerdo a su presupuesto; en un lugar se accede con el presupuesto ≤ 200,000 y el otro se accede con el presupuesto > 200,000.

Los predicados simples para la primera consulta serían:
p_1 : Localización = "México"
p_2 : Localización = "Monterrey"
p_3 : Localización = "Puebla"

Los predicados simples para la segunda consulta serían:
p_4 : Presupuesto ≤ 200,000
p_5 : Presupuesto > 200,000

Si el algoritmo COM_MIN es seguido, el conjunto Pr' = {p_1, p_2, p_3, p_4, p_5} es obviamente completo y mínimo. Basado en Pr', los siguientes seis minitérminos que forman a M se pueden definir como:

m_1: (Localización = "México") ∧ (Presupuesto ≤ 200,000)
m_2: (Localización = "México") ∧ (Presupuesto > 200,000)
m_3: (Localización = "Monterrey") ∧ (Presupuesto ≤ 200,000)
m_4: (Localización = "Monterrey") ∧ (Presupuesto > 200,000)
m_5: (Localización = "Puebla") ∧ (Presupuesto ≤ 200,000)
m_6: (Localización = "Puebla") ∧ (Presupuesto > 200,000)

Estos no son los únicos minitérminos que se pueden generar. Por ejemplo, es posible especificar predicados de la forma:

$p_1 \wedge p_2 \wedge p_3 \wedge p_4 \wedge p_5$

Sin embargo, las implicaciones obvias:
i_1: $p_1 \Rightarrow \neg\, p_2 \wedge \neg\, p_3$
i_2: $p_2 \Rightarrow \neg\, p_1 \wedge \neg\, p_3$
i_3: $p_3 \Rightarrow \neg\, p_1 \wedge \neg\, p_2$
i_4: $p_4 \Rightarrow \neg\, p_5$
i_5: $p_5 \Rightarrow \neg\, p_4$
i_6: $\neg\, p_4 \Rightarrow p_5$
i_7: $\neg\, p_5 \Rightarrow p_4$

eliminan esos minitérminos y nos quedamos con m_1 hasta m_6. Observando la instancia de la base de datos del ejemplo, podríamos decir que las siguientes implicaciones se mantienen:

i_8: (Localización = "México") ⇒ ¬ (Presupuesto > 200,000)
i_9: (Localización = "Puebla") ⇒ ¬ (Presupuesto ≤ 200,000)
i_{10}: ¬ (Localización = "México") ⇒ (Presupuesto ≤ 200,000)
i_{11}: ¬ (Localización = "Puebla") ⇒ (Presupuesto > 200000)

Sin embargo, recuerde que las implicaciones deben ser definidas de acuerdo a la semántica de la base de datos y no de acuerdo a los valores actuales. Algunos de los fragmentos definidos por M = { m_1, m_2, m_3, m_4, m_5, m_6 } pueden estar vacíos, pero ellos son, no obstante, fragmentos. No existe nada en la semántica de la base de datos que sugiera que las implicaciones i_8 hasta i_{11} se satisfagan.

El resultado de la fragmentación horizontal primaria de la relación *Proyecto* forman seis fragmentos $F_{Proyecto}$ = {Proyecto$_1$, Proyecto$_2$, Proyecto$_3$, Proyecto$_4$, Proyecto$_5$, Proyecto$_6$} de acuerdo a los minitérminos de *M*. Algunos de esos están vacíos (Proyecto$_2$, Proyecto$_5$) y por lo tanto no se presentan aquí.

Proyecto$_1$

noProyecto	Nombre	Presupuesto	localización
P1	Instrumentación	150,000	México

Proyecto$_3$

noProyecto	Nombre	Presupuesto	localización
P2	Desarrollo de BD	135,000	Monterrey

Proyecto$_4$

noProyecto	Nombre	Presupuesto	localización
P3	Diseño asistido por computadora.	250,000	Monterrey

Proyecto$_6$

noProyecto	Nombre	Presupuesto	localización
P4	Mantenimiento.	310,000	Puebla

Fragmentación horizontal primaria de la relación PROYECTO

Procesamiento y Optimización de Consultas

La optimización y procesamiento de consultas, implica técnicas los cuales los SGBD: *Procesan Optimizan y Ejecutan consultas de alto nivel*.

Expresar una consulta de alto nivel como SQL, implica:
1. *Examen o análisis léxico*: Verificar los componentes del lenguaje, en este caso SQL.
2. *Análisis Sintáctico*: Consiste en revisar la sintaxis para determinar si la consulta está formulada de acuerdo con las reglas sintácticas, es decir las reglas gramaticales.
3. *Validación*: Los nombre de atributos y de las relaciones sean válidas.

El procedimiento es el siguiente, se crea una representación interna de la consulta (estructura en forma de árbol o grafo, llamado: *árbol o grafo de consulta*).

Se crea una estrategia de ejecución en el SGBD para obtener el resultado de esa consulta a partir de los archivos internos de la base de datos (repositorio de datos).

El proceso de elegir la más adecuada estrategia de ejecución para procesar una consulta, es conocido como: *Optimización de consulta*.

Pasos del procesamiento de una consulta de alto nivel (Ver Figura 5).

El SGBD evalúa sistemáticamente estrategias alternativas de ejecución de una consulta y escoge la óptima.

El SGBD cuenta con algoritmos generales de acceso a la BD que implementan operaciones relacionales como la selección, reunión o sus combinaciones.

El módulo de optimización de consultas, sólo considera las estrategias de ejecución que ponen en práctica los algoritmos de acceso del SGBD y se apliquen al diseño particular de la BD.

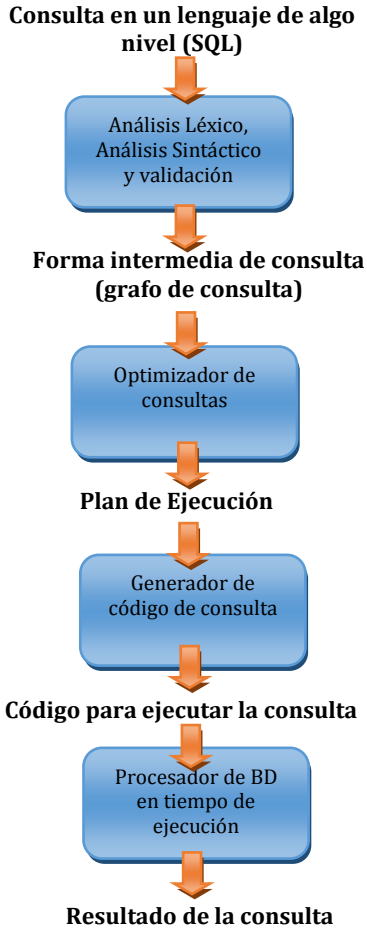


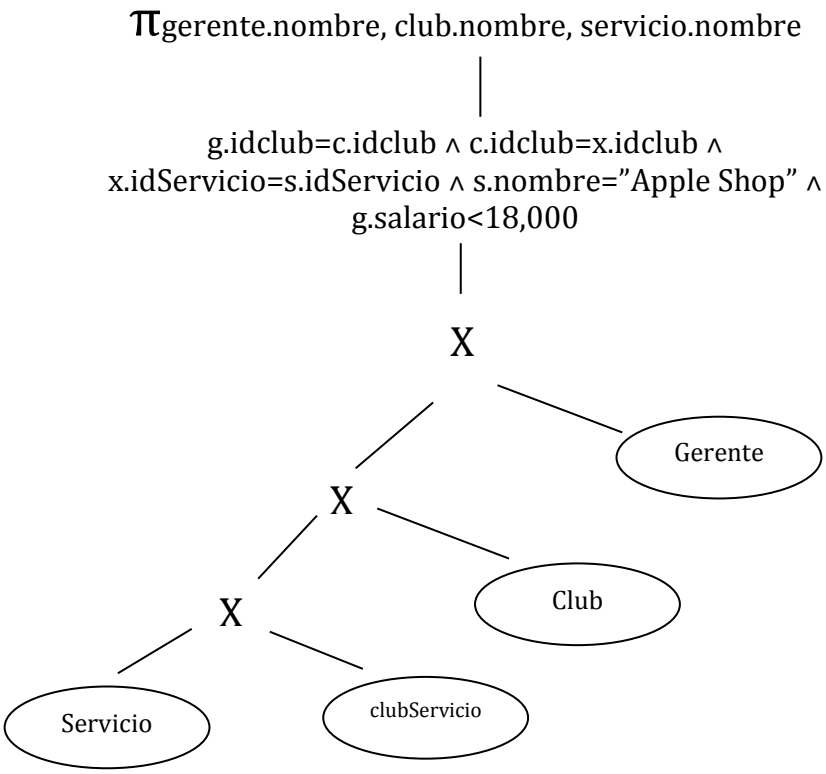
Figura 5. Pasos del procesamiento de consultas

Empleo de la heurística en la optimización de consultas

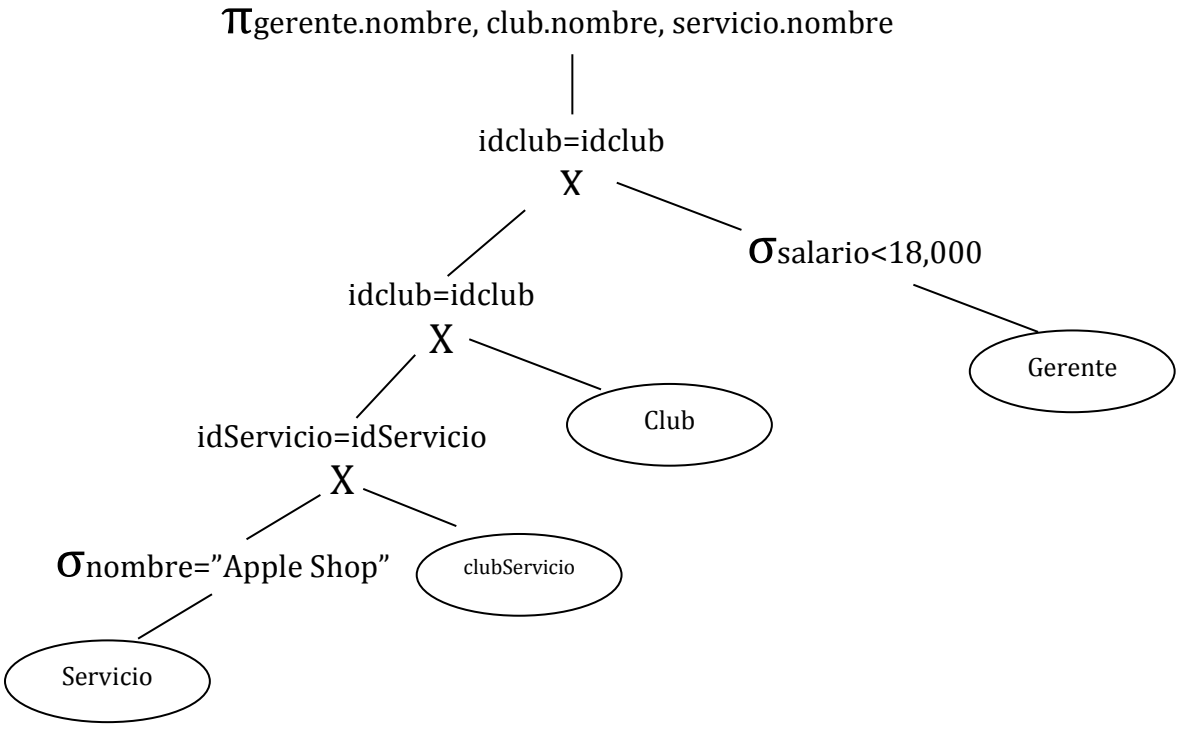
Considerando la siguiente consulta:

```
SELECT g.nombre, c.nombre, s.nombre
FROM Gerente g, Club c, Servicio s , clubServicio x
WHERE g.idclub=c.idclub
AND c.idclub=x.idclub
AND x.idServicio=s.idServicio
AND s.nombre="Apple Shop"
AND g.salario<18,000;
```

a) Construir el árbol canónico

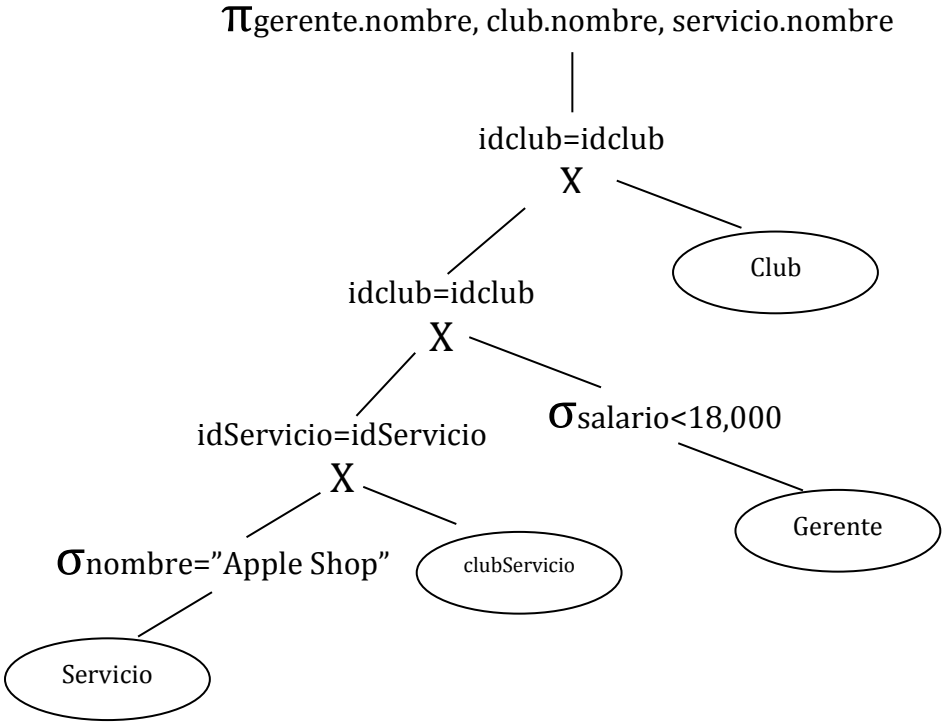


b) Construir el árbol de consulta (π)



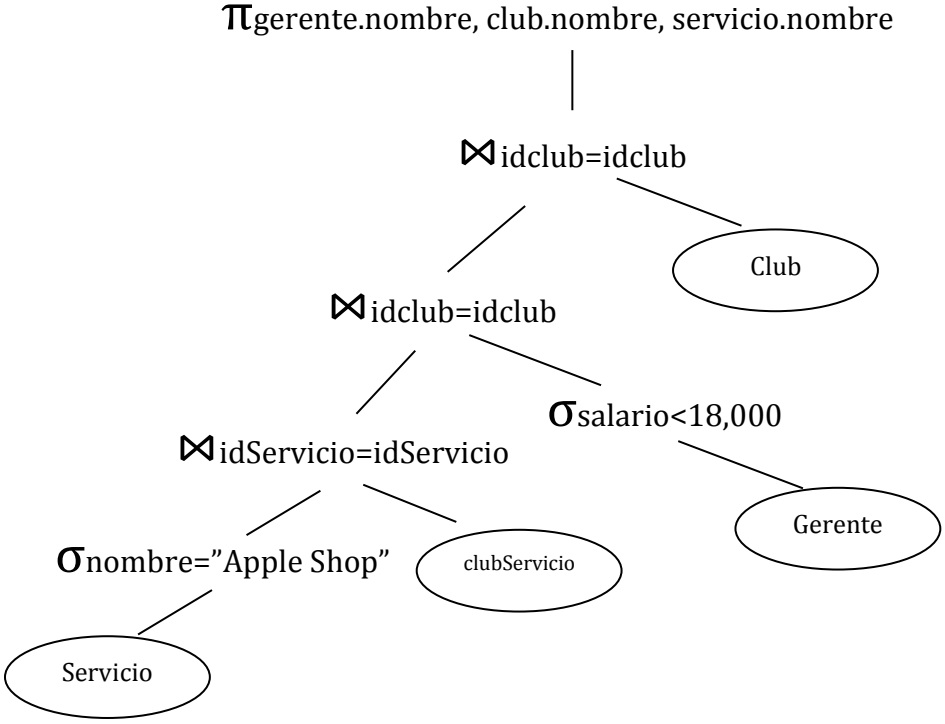
c) Construir el árbol de consulta (X)

Aplicar si es necesario la propiedad de conmutación entre las hojas; tener en el nivel inferior aquellas relaciones que tienen el menor número de tuplas.



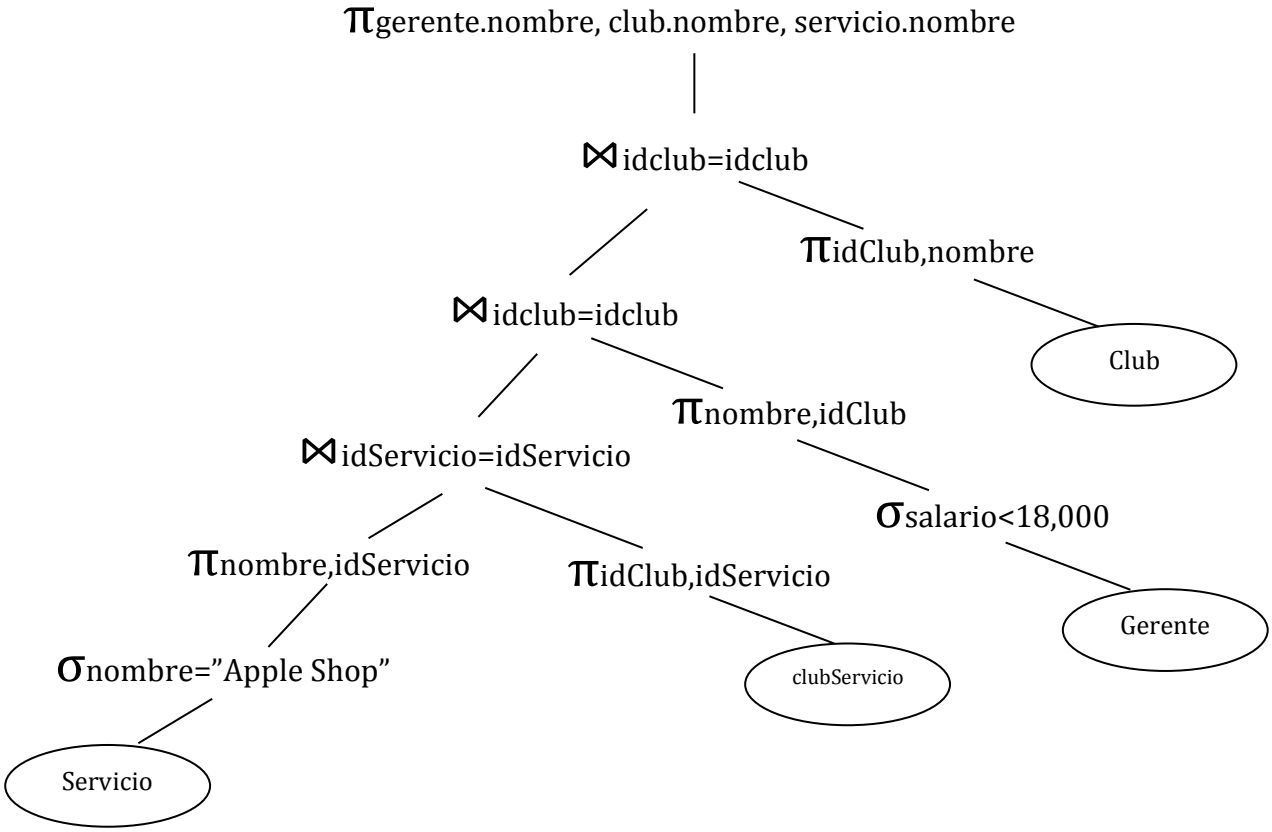
d) Construir el árbol de consulta (⋈)

Cuando se existen atributos de reunión y el producto cartesiano, se deben sustituir por el operador de reunión (⋈).



e) Construir el árbol de consulta (π)

Proyectar únicamente aquellos atributos que aparecen en el nodo raíz y los que son necesarios para llevar a cabo la operación de reunión, excluir aquellos atributos que inclusive se establezcan condiciones de selección excepto si son indicados en el nodo raíz.



f) Generar Expresiones Algebraicas

$R_1 \leftarrow \pi_{nombre,idServicio}(\sigma_{nombre="Apple\ Shop"}(Servicio))$

$R_2 \leftarrow \pi_{idClub,idServicio}(clubServicio)$

$R_3 \leftarrow \pi_{nombre,idClub}(\sigma_{salario<18,000}(Gerente))$

$R_4 \leftarrow \pi_{idClub,nombre}(Club)$

$R_5 \leftarrow R_1 \bowtie_{idServicio=idServicio} R_2$

$R_6 \leftarrow R_5 \bowtie_{idClub=idClub} R_3$

$R_7 \leftarrow R_6 \bowtie_{idClub=idClub} R_4$

$\pi_{gerente.nombre,\ club.nombre,\ servicio.nombre}(R_7)$