ESTRATEGIA DE PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS EN UN GESTOR DE BASES DE DATOS FEDERADAS

Agustín J. García Banda
Dept. de Llenguatges y Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona, España
Septiembre, 2001
ajgarcia@lsi.upc.es

Abstract

Based on **BLOOM** (**B**arce**L**ona **O**bject-**O**riented **M**odel), we propose to establish a global query processing strategy in the Federated Query Manager. First, the developing mechanism constructs a tree, which has nodes that initially represent explicit joins between federated classes and that are decomposed, into implicit joins between classes in the Component Schemas. Consequently, different heuristic techniques are performed in order to optimise the decomposition process, which generate one or more Execution Plans (EP). After that, the EP are analysed to get the optimum. The objective function of this strategy is to choose an execution plan with the least total resource usage and the best response time. Finally, the consolidation of partial results is carried out maintaining the federated result in the root node.

Resumen

Basado en el Proyecto **BLOOM** (**B**arce**L**ona **O**bject-**O**riented **M**odel), se propone establecer una estrategia de procesamiento de la consulta global dentro del Gestor de Consultas Federado. Primero, se construye un árbol cuyos nodos inicialmente representan joins explícitos entre clases federadas y los cuales son descompuestos en joins implícitos entre clases en los esquemas componentes. Posteriormente, diferentes técnicas heurísticas optimizan el proceso de descomposición, las cuales generan uno o mas Planes de Ejecución (PE). Después, éstos PE son analizados para obtener el optimo. La función objetiva de esta estrategia es encontrar un plan de ejecución con el menor uso de recursos y el mejor tiempo de respuesta. Finalmente, la consolidación de resultados parciales se lleva a cabo manteniendo la respuesta federada en el nodo raíz.

Contenido

1. Introducción

Diferentes tipos de ***B**ases de **D**atos* (**BDs**) han sido desarrolladas e implementadas para estar a la altura de las demandas de los usuarios. Éstas BDs pueden ser diseñadas independientemente en una organización. Como resultado , la heterogeneidad de las BDs es una situación inevitable cuando diferentes tipos de BDs coexisten en una organización que trata de compartir datos entre éstas. Por lo que muchos investigadores han enfocado sus esfuerzos en la exploración de un esquema global que trate de resolver el problema de la heterogeneidad de las BDs. Para esto se han propuesto muchos enfoques de integración como en [BE-L96], Estos enfoques usan un ***S**istema de **B**ase de **D**atos **F**ederado* (**SBDF**) [SL-P90], para soportar la interoperabilidad de las BDs heterogéneas. Los SBDF emplean esquemas federados únicos o múltiples para resolver la heterogeneidad de esquemas entre las BDs locales. Con un esquema federado, los SBDF pueden soportar la interoperabilidad de las fuentes de información integrando los esquemas locales.

Cuando la federación ha sido construida, se desarrolla una nueva arquitectura diferente llamada “Arquitectura de Ejecución”, en la cual se crean capas de software (módulos) que gestionan el flujo de datos y procesan la consulta global. Esto permite que subconsultas se ejecuten sobre las **B**ases de **D**atos **C**omponentes (**BDCs**) donde la información es extraída. Finalmente, se combinan resultados parciales para constituir un único resultado, ver figura 2. Existen investigaciones que han implementado una metodología o estrategia para llevar a cabo la consolidación de resultados y obtener una respuesta federada, pasando por la descomposición de la consulta, como se menciona en [IEU-P94, KC-P95, LPL-P96, LSriP93, YW-L97, WC-L95]

El procesamiento de consultas eficientes en sistemas *Multidatabase* distribuidas y heterogéneas es un problema importante para el futuro desarrollo de sistemas de información global [SL-P90], por lo tanto, el mejor desempeño del procesamiento de consultas en sistemas federados dependen principalmente de la optimización de los *Planes de Ejecución* de la consulta global [Du-P92, GeSe-P99].

Varias son las técnicas empleadas para optimizar una consulta federada antes y después de la descomposición de la misma consulta, estos procesos algorítmicos son sustentados tanto para sistemas de gestión de BDs heterogéneas con esquemas federados relacionales [GeSe-P99,WC_L95, Fynn_P97, LOG_P98, HK_P99], como para esquemas Orientados a Objetos [Asumap97, ON_P97, BE-L96, GST_P96], que es una de las finalidades de este documento en particular.

En éste trabajo se propone una estrategia de procesamiento, en la cual, se diseñan e implementan los submódulos de **Descomposición de Consultas** y **Consolidación de Resultados** dentro del Gestor de Consultas Federadas. El mecanismo de desarrollo establece la construcción de un árbol cuyos nodos se descomponen inicialmente en *join explícitos* y posteriormente en *join implícitos*, donde las clases atómicas permanecen contra los esquemas componentes. Hecho lo anterior, algoritmos son aplicados a las subconsultas intermedias sobre predicados y proyecciones. Una vez ejecutado los algoritmos que cubren el proceso de descomposición, entra en función diferentes técnicas de optimización, afectando a los dos submódulos. Diferentes **técnicas heurísticas** se presentan para optimizar la descomposición basada en una búsqueda exhaustiva de planes de ejecución, empezando con predicados presentados y posteriormente un proceso que realiza una **estimación del costo** de los planes de ejecución. Éste apoyado por estadísticas de las BDCs, las cuales se procesan con un **modelo del costo**. Finalmente, la consolidación de resultados parciales se lleva a cabo manteniendo la respuesta federada en el nodo raíz.

Como función objetiva, se busca optimizar los recursos y el tiempo de respuesta, por lo tanto, se pretende buscar el mejor plan de ejecución con el mínimo costo de recursos y menor tiempo de respuesta.

En éste trabajo se toma en cuenta el *esquema integrado* del BLOOM (basado en la semántica) y *diccionario de datos* que mantiene las correspondencias entre clases entre otras. Dentro de los artículos [SCI-P99, AORS99, GSCP95] todos los puntos de interés son tomados en cuenta, resaltando los siguientes:

- El Modelo de Datos Canónico es Orientado a Objetos [ASS-P99, SCGP91]
- Para la Integración de Esquemas utiliza una metodología llevada a cabo por la fase de enriquecimiento semántico [CAS-P93], la fase de detección [GSCP93] y la fase de resolución, para mas detalle ver en [GSCP95].
- Basado en un lenguaje declarativo orientado a objetos para el procesamiento de consulta global en el modelo de datos orientado a objetos, es decir una consulta que consiste de tres partes: la cláusula Select, From y Where, el formato de una consulta es el siguiente; Select <*atributo objeto*>, From <*serie de clases*> y Where <*cláusula de predicados*> .
- El árbol de ejecución es creado en el módulo de descomposición, el cual empieza a formular un plan inicial de ejecución de la consulta. Este plan es construido de una manera ordenada. El módulo de descomposición de la consulta es también responsable de establecer las correspondencias apropiadas del esquema global a los esquemas locales [AGF-P95].
- El directorio de datos es de suma importancia para definir correspondencias entre instancias de atributos, objetos en una clase virtual y una clase constituyente. La **discriminación de generalización** y la **función de identificación de objetos (oif)** presentados en [GSCP95] son tomados en cuenta.

Éste documento esta ordenado de la siguiente manera: En la sección 2 y 3 se presentada una arquitectura de construcción y ejecución, respectivamente (Proyecto BLOOM). En la sección 4 se describe la estrategia de descomposición de la consulta federada y su consolidación de resultados son mostrados en la sección 5. En la sección 6 entramos de lleno con la Optimización de la consulta global, en la cual, el punto 6.1 ilustra la heurística tomada para la fase 2, y el punto 6.2 el plan de ejecución final que responde a las técnicas de estimación del costo de planes de ejecución, y posteriormente se demuestra la selección de la mejor estrategia, así como, la aplicación de la técnica de paralelismo que apoya el mejor tiempo de respuesta. Finalmente, en la sección 7 presentamos las conclusiones del trabajo.

2. Arquitectura de Construcción

Construir un sistema que soporte el acceso integrado es de suma importancia. Por lo que antes que un SBDF opere, se tienen que resolver diferentes problemas relacionados con las heterogeneidades, tal como, la heterogeneidad sintáctica y semántica, así como, los problemas de distribución. Por esta razón, el SBDF debe tener una adecuada arquitectura de esquema y un conveniente modelo de datos global llamado ***M**odelo de **D**atos **C**anónico* (**MDC**) (ver figura 1), referida al [ROSC-P97, SL-P90]

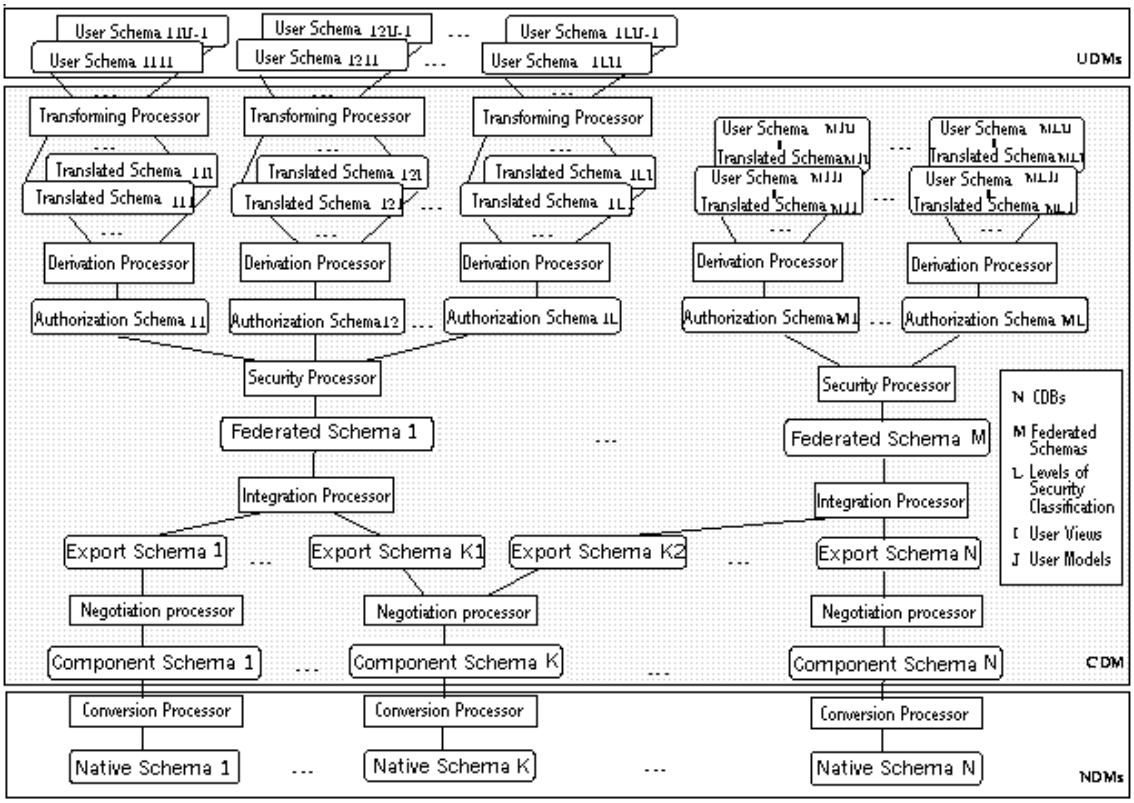


Fig. 1. Arquitectura de construcción del BLOOM

3. Arquitectura de Ejecución

El flujo de datos empieza con una consulta planteada por el usuario de la federación llamada *consulta federada*. El proceso por módulos es el siguiente, propuesto en [ROSC-P97] (ver figura 2):

1) **Controlador de Seguridad**: Pasa por el módulo llamado *Controlador de Seguridad*, autorizando la consulta dada.
2) **Transformador de Consultas**: Continua con el Transformador de Consultas generando una consulta equivalente expresada en términos del ***E**squema **F**ederado* (**EF**)y etiquetado con el nivel de seguridad *nv*.
3) **Descomposición de Consultas (Query Decomposer, QD)**: El módulo Descomposición de Consultas es el responsable de descomponer la consulta global en subconsultas contra las BDCs.
4) **Traductor de Consultas**: Posteriormente el Traductor de Consultas da una subconsulta equivalente en el modelo de datos nativo de la BDC expresado en términos del correspondiente ***E**squema **N**ativo* (**EN**)y etiquetado con un nivel de seguridad federado *nv*.

5) **Traductor de Nivel de Seguridad**: El Traductor del Nivel de Seguridad etiqueta las subconsultas con el nivel de seguridad componente *nv* de su BDC que corresponde a ese nivel federado. Ahora, se esta en condiciones de ejecutar las correspondientes consultas bajo el control del *gestor de transacciones*.
6) **Traductor de Subresultados**: Una vez que se tiene la consulta hecha, los resultados llegan al Traductor de Subresultados, donde son convertidos al MDC.
7) **Consolidador de Resultados (Result Consolidator, RC):** Se produce el resultado final en BLOOM, usando las correspondencias de los Esquemas Federados y **E**squemas **C**omponentes (**ECs**).
8) **Transformación de Resultados**: El módulo de Transformación de Resultados usando las correspondencias del Esquema Federado y el Esquema del Usuario, presenta el resultado final en el modelo del usuario, expresado en términos de su Esquema del Usuario.

El punto número 3 y 7 son investigados y detallados. Como continuación se propone una optimización para el punto 3, el cual será analizada y presentada en este documento.

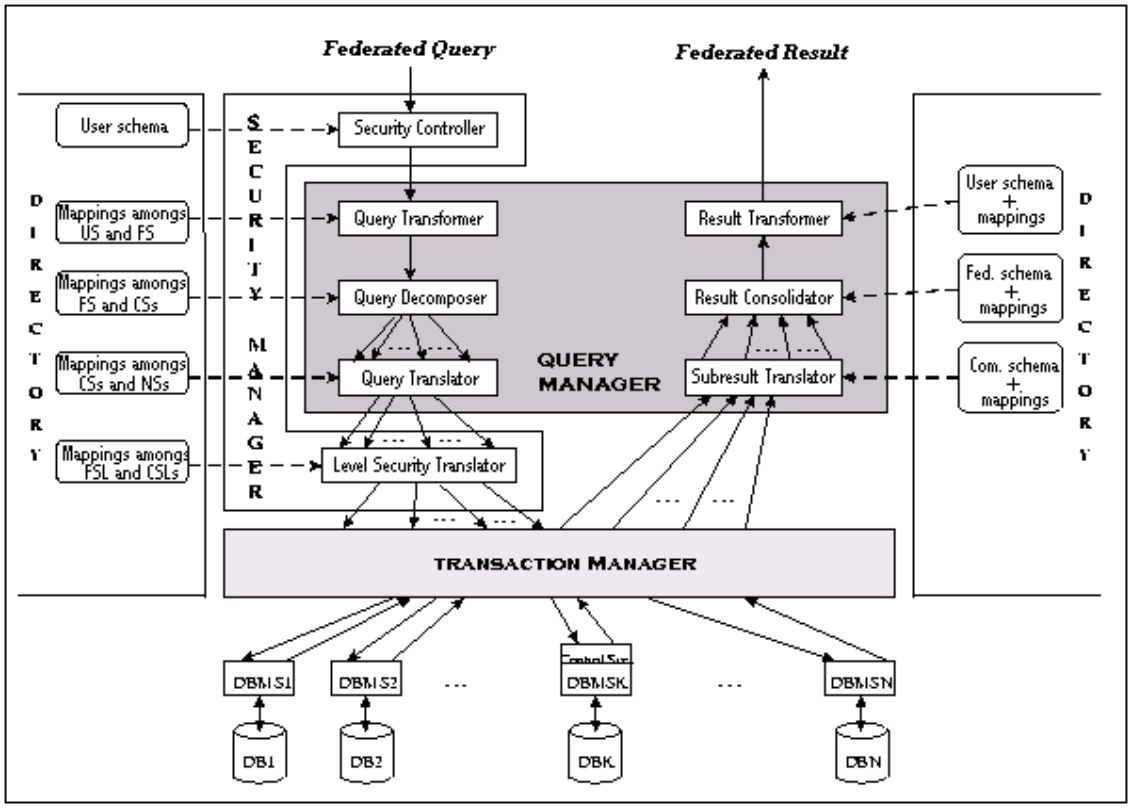


Fig. 2. Arquitectura de ejecución del BLOOM

4. Descomposición de la Consulta Federada

4.1 Metodología de Descomposición:

Una vez que el Transformador de Consultas ha generado una consulta equivalente en términos del EF mencionado en el punto 3 de la Arquitectura de Ejecución, el módulo de descomposición de consultas obtiene como entrada una consulta global con clases virtuales, las cuales fueron creadas por el proceso de integración de esquemas después de aplicar los operadores de integración mencionados en [GSCP95], contando con las respectivas correspondencias de sus clases virtuales almacenadas en el Directorio.

Después del punto anterior, la consulta global puede mantener datos referenciados de más de una BD; Por lo que ninguna de estas consultas puede ser evaluada directamente por una BDC, es por consiguiente, que se tiene que crear un algoritmo (parte de la capa de software) que apoye a la descomposición y consolidación de la consulta global. A continuación se describen los puntos estratégicos a tomar en cuenta para iniciar la descomposición.

Una consulta global consiste de tres partes importantes (Cláusula Select, From y Where) y el formato es el siguiente para éste caso en particular:

Select <*Atributos*>
From <S*erie de clases*>
Where <*Predicados*>

El módulo encargado de la descomposición de la consulta global, se divide a su vez en 3 fases relevantes, la primera inicia inmediatamente después del proceso que ejecuta el módulo de transformación de la consulta, por lo que se tiene como entrada a una consulta equivalente expresada en términos del EF y etiquetada con un nivel de seguridad, La fig. 3, muestra las fases del QD. La primera fase es un **Parser y Modificación de la Consulta Global**, la cual analiza la carga de clases que ésta mantiene y que obtiene directamente de la cláusula *From,* con lo cual construye un árbol para *joins explicitos*; la segunda fase es la más importante ya que en ella se examina cada clase virtual y posteriormente se detectan los Esquemas Componentes, a fin de determinar la correspondencia entre Esquemas Componentes y Clases Virtuales, lógicamente esto se debe a la integración de esquemas realizado en la fase de resolución, es aquí donde tomaremos en cuenta el **atributo de discriminación** propuesto en [GSCP95].

Una vez hecho lo anterior, entra el proceso de **Descomposición en Subconsultas** dependiendo de los ECs participantes, es aquí donde el número de subconsultas permanece igual al número de ECs, pero con diferencias en su cláusula *From* donde contiene ahora una expresión de mapeo que corresponde al EC.

Después se procesa cada subconsulta y se detectan las clases que componen la clase virtual, formando un número de subconsultas igual a las clases que forman la clase virtual. Cuando el conjunto de nodos se haya reunido, estos estarán en el nivel más bajo del árbol (nodos hojas del árbol) y se aplicará un proceso donde la participación de las cláusulas *Where* y *Select* toman un matiz importante (refiriéndose a los predicados y proyecciones). Se procesan los predicados y se depuran las consultas, más adelante se explica el proceso de depuración. Finalmente en la tercera fase se detectan las BDCs y se etiquetan las consultas para su regreso y buen funcionamiento del módulo encargado de la consolidación de resultados.

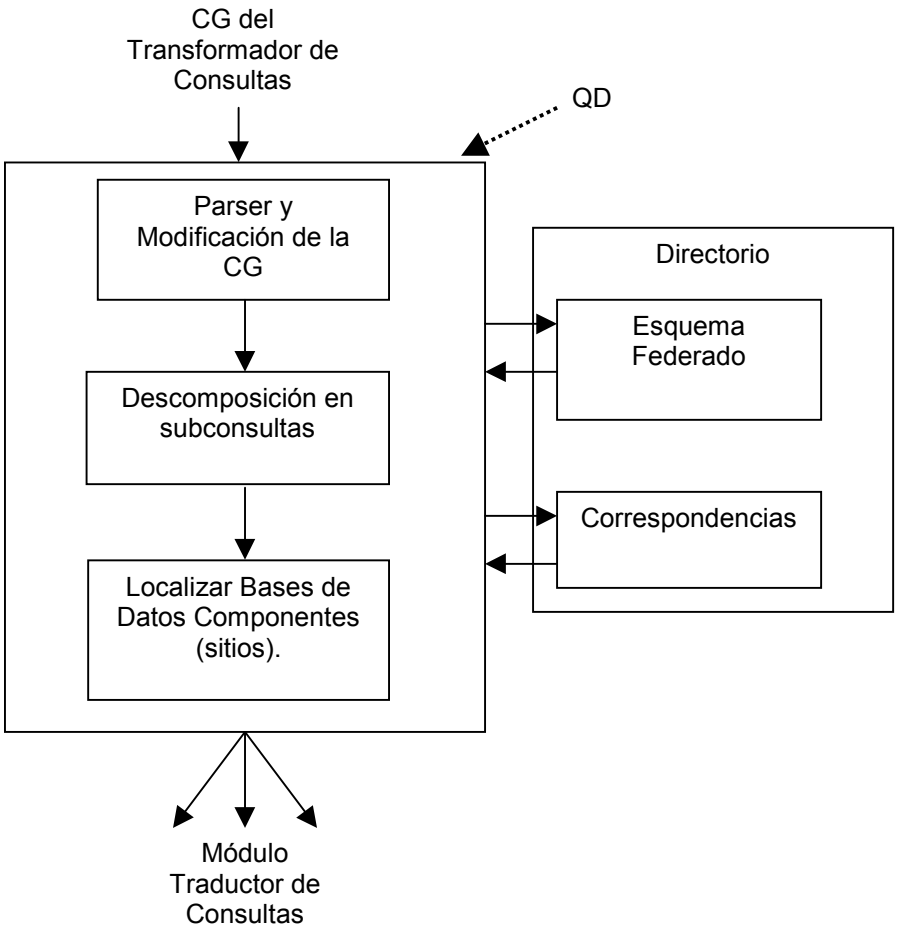


Fig. 3. Módulo de Descomposición de Consultas Globales, (Query Decomposer –QD-)

Fase 1. Parser y Modificación de la Consulta Global

Analiza si más de una serie de clases están especificadas en la cláusula *From* para sus *joins explícitos* (*explicit joins*), si es así, la consulta global debe ser descompuesta en una colección de subconsultas, las cuales contienen solo una serie de clase. Lo anterior da lugar a una consulta global modificada para procesar los *joins,* el objetivo es integrar los resultados de las subconsultas (Creación del árbol de la consulta global).

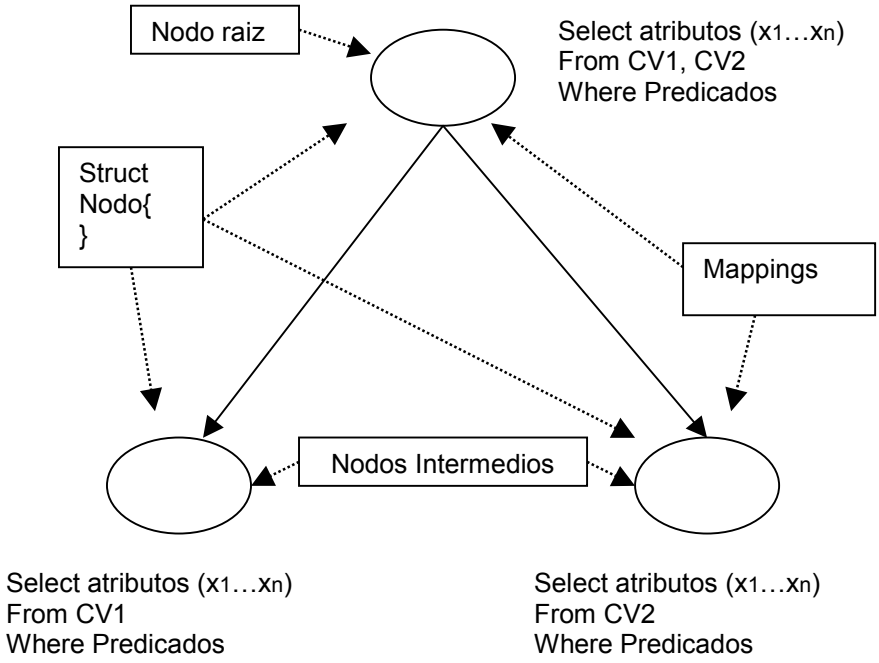
========== Algoritmo 1 ==========

```
/* Este algoritmo (pseudocódigo) crea una o más consultas modificando la consulta global. El
cual toma en cuenta el número de clases virtuales (joins explicitos) para realizar la
descomposición. */
/* Declaración de Variables
CVJ[]:=0
struct NodoRaiz []{                          /* El árbol sintáctico comprende un nodo raíz */
        Inf := Q                             /* Consulta Global a examinar */
        Resp := set (objetos)                /* Respuesta Consolidada */
}
Struct Nodo [] {
        Padre := NodoRaiz[]                  /* Los nodos que contienen las subconsultas
                                             apuntan al padre para efectos de la
                                             consolidación*/
        Inf := Q
```

```
        Nivel := n
        Resp := set(objetos)
        Pos := n
}
Procedure Tree_Sint(CG[], CVJ[], TCVJ)      /* Consulta Global, Modificada y conjunto de
                                             clases variable*/
Begin
        Integer: C_1:=1
        for (C_1, C_1<=TCVJ, C_1++)
                New Nodo[]                   /* Nuevo nodo */
                Nodo[].Pos := C_1            /* Da posición del nodo
                FromTemp:=CVJ[C_1]           /* Obtiene clase virtual */
                P_Modi_Con(CG[], FromTemp, CG_M[])      /* Procedimiento que modifica
                                             la Consulta Global, y construye una consulta
                                             solo con la cláusula virtual C_1 */
                Nodo[].Inf := CG _M          /*Anexa la consulta global modificada con
                                              clase virtual única */
                FromTemp= []                 /* Conjunto vacio */
         end_for (TCVJ)
end_procedure

Begin (EI, Map)                              /*Parametros: Esquema Integrado, Mappings

/* Dada una Consulta Global de Entrada
CG[] = get (CG_OQL[])                        /* Obtener Consulta Global
get_clase (CG_OQL[], CVJ[])                  /* Obtener clases virtuales (joins Exp)
TCVJ=CtoI(CVJ[])                             /* Número de clases virtuales
   if  (TCVJ > = 1)
     then
        Procedure Tree_Sint(CG[],  CVJ[], TCVJ)      /*Descompone en
                                                      clases virtuales únicas
    end_if (TCVJ)

end_Begin
```

========== Fin del Algoritmo 1 ==========

La explicación del algoritmo es el siguiente: Dada una consulta global, se toma en cuenta el esquema integrado, realizado por las fase de detección y resolución, con estas fases ya terminadas se crean también los *mappings* (correspondencias). Una vez todo esto, se resuelve el número total de clases virtuales (si son dadas) y se envía a un procesamiento para generar un árbol, en donde las subconsultas quedan registradas en forma de nodos con una clase virtual única, por lo que ahora se envía a la siguiente fase para analizar cada una de éstas, y seguir con la descomposición. Para aclarar este punto se menciona un ejemplo abstracto en la figura 4.



CV: Clase Virtual

Figura 4. Parser y modificación de la Consulta Federada, fase 1.

Fase 2. Descomposición en subconsultas.

Un punto a considerar después de la primera fase, es que, a pesar de que una consulta modificada contenga una clase virtual global, ésta pueda ser una clase virtual de la forma CV={*C1,...,Cn*} donde *C* es una clase implícita de la CV, la cual corresponde a una BDC. En esta situación la clase virtual es el resultado de una operación de integración (en BLOOM, las dimensiones de agregación y generalización son base de su integración), y por tal motivo se construye un algoritmo dentro de este submódulo que descompone cada consulta modificada, la cual fue sometida a la fase 1, y que ahora mantiene una clase virtual única. Esta descomposición es apoyada por operadores de correspondencias, controlados por un *mappings* (creación de un algoritmo para las correspondencias).

========== Algoritmo 2 ==========

/* Este algoritmo descompone una consulta modificada con una *clase virtual única* en subconsultas que se presentan cada una contra su EC respectivo, generando nodos intermedios. Posterior a este proceso y tomando en cuenta que la clase virtual consistente de varias clases de diferentes esquemas componentes, crea nodos hojas sin reestructurar con clases componentes. Finalmente con la depuración de estas obtenemos nodos reestructurados */

```
/* Declaración de Variables:
Struct Nodo [] {
        Padre := NodoRaiz[]          /* Los nodos que contienen las subconsultas
                                        Apuntan al padre para efectos de la consolidación*/
        Inf := Q
        Nivel := n
        Resp := set(objetos)
        Pos := n
```

```
}
Struct Nodo2 [] {
        Padre := Nodo[]
        Inf := Q
        Nivel := n
        Resp := set(objetos)
        Pos := n
}

 Struct Nodo3 [] {
        Padre := Nodo2[]
        Inf := Q
        Nivel := n
        Resp := set(objetos)
        Pos := n
}

Nodo2[],Nodo3[]:= []
EC = n                                          /* número de esquemas componentes
Begin (Nodo[], TCVJ)                            /* toma como entrada de la consulta
                                                   global, Nodos del árbol sintácticos y
                                                   Clases Virtuales */
/* Entrada de una subconsulta modificada de fase 1
C_2 := 1
for (C_2, C_2<=TCVJ, C_2++) do                  /* Ciclo de análisis de cada subconsulta
        seek Nodo[](Nodo[].Pos == C_2)          /* Apunta al  Nodo correspondiente */
        Procedure Get_Clase_Unica (Nodo[] , CV)         /* Obtiene la clase única de la
                                                           Subconsulta. */
        Procedure Get_Examina(CV, NumEsqC[])    /*  Procedimiento que examina número
                                                   de esquemas componentes que se
                                                   involucran para ésta clase virtual */
        TnumEsqC := CtoI(NumEsqC[])             /* Obtiene número de EC */

        for (C_3:=1, C_3<=TNumEsqC, C_3++)      /* Ciclo entrada a Nodos */
                New Nodo2[]                     /* Nuevo Nodo */
                Nodo2[].Padre := C_2
                Nodo2[].Pos := C_3
                Procedure Corr_EC(Nodo[], Nodo2[])  /* Procedimiento que aplica
                                                       correspondencia (mapping) contra el
                                                       EC, generando un Nodo en cada EC
                                                       con la clase virtual C_1 de la fase 1
                                                       modificada con sus clases atómicas
                                                       implícitas. (joins implícitos)  */

        end_for (TNumEsqC)

        for (C_4 := 1, C_4<=TnumEsq, C_4++)     /* Ciclo de reconocimiento de
                                                   esquemas comp. */
                seek Nodo2[](Nodo[].Pos == C_4) /* Busca Nodo

                C_EC[] := Nodo2[].Inf           /* Obtiene subconsulta contra cada EC

                Procedure Get_Examina(C_EC[], NumClasesC[]) /* Procedimiento que
                                                       examina el número de clases
                                                       componentes para cada subconsulta
                                                       que cuenta con una clase virtual única.
                                                       */
                TnumClasesC := CtoI (NumClases[])   /* Total de clases
                                                       componentes de la clase virtual */
                for (C_5 := 1, TnumClasesC > C_5, C_5++)
```

```
                                        /* Ciclo de reconocimiento de clases
                                        componentes o participantes de cada
                                        clase virtual*/
            New Nodo3[]                 /* Nuevo Nodo
            Nodo3[].Padre := C_4
            Nodo3[].Pos   := C_5
            Procedure Coloca_ClaseComp(NumClases[C_4], Nodo3[])
                        /* Crea nodo semi-final con clases
                        exclusivamente atómicas, aquí solo pone la
                        clase correspondiente en la subconsulta para
                        Nodo3[].inf */
            Procedure Etiqueta_oif [C_4], Nodo3[])  /* En este procedimiento
                        reconoce los oif dentro de la estructura
                        semántica, junto con las aserciones
                        semánticas (E-SR y S-SR [GSCP95]) */
        end_for (TnumClasesC)
    end_for (TnumEsqC)

    for (C_6 := 1, TnumEsqC, C_6++)             /* Ciclo de reconocimiento de
                                                subconsultas contra EC */
        for (C_7 := 1, TnumClasesC, C_7++)      /* Ciclo de reconocimiento de
                                                clases atómicas de cada clase
                                                virtual*/
            seek Nodo3[] (Nodo3[].Padre == C_6  && Nodo3[].Pos == C_7)
                                                /* Busca Nodo, se analizan
                                                nodos finales por lo que ven
                                                todas las subconsultas */
            Procedure Analiza_predicados (Nodo3[])      /* las cláusulas Where
                                                son analizadas y toman en
                                                cuenta sus operadores binarios
                                                */
            if (Nodo3[] == factible)
               then
                  Procedure Analiza_select (Nodo3[])  /* Ve si atributos pertenecen

                  Procedure Analiza_path_expresión (Nodo3[]) /* sustituye por
                                                atributo materializado */
                  ..........                    Procedure
Analiza_standby (Nodo3)[] /* Nodo que necesita
                                                información de otros nodos
                                                para ejecutarse */


               else
                     Procedure Elimina (Nodo3[])    /* Elimina esta subconsulta que
                                                redunda con las otras */
                     Procedure Corrimiento_nodos (Nodo3[]) /* debido al espacio de

                                                eliminación, es preciso hacer
                                                una reordenación de nodos */
                     Loop;                      /* continua con otro Nodo */

            Nodo3[] := Nodo restructurado + Oid
                                                */ Estos nodos reestructurados
                                                necesitan un Oid, que se
                                                anexan a la cláusula Select,
                                                para su identificación en la
                                                consolidación de resultados */
            end_if (Nodo3[])
```

```
                end_for (TnumclasesC)
        end_for (TnumEsqC)
end_for (TCVJ)
/*salida  */
end_Begin
```

========== Fin del Algoritmo 2 ==========

El algoritmo para la fase de descomposición se explica de la siguiente manera:
Dada una Consulta Global con una clase virtual o una Consulta Modificada (obtenida de la fase 1) se examinan las correspondencias de la clase para determinar el número de clases contra esquemas componentes que mantiene la clase virtual ya sea de la consulta global o de la consulta modificada, esto se apoya directamente con los **atributos de discriminación** formulados en la integración de esquemas.

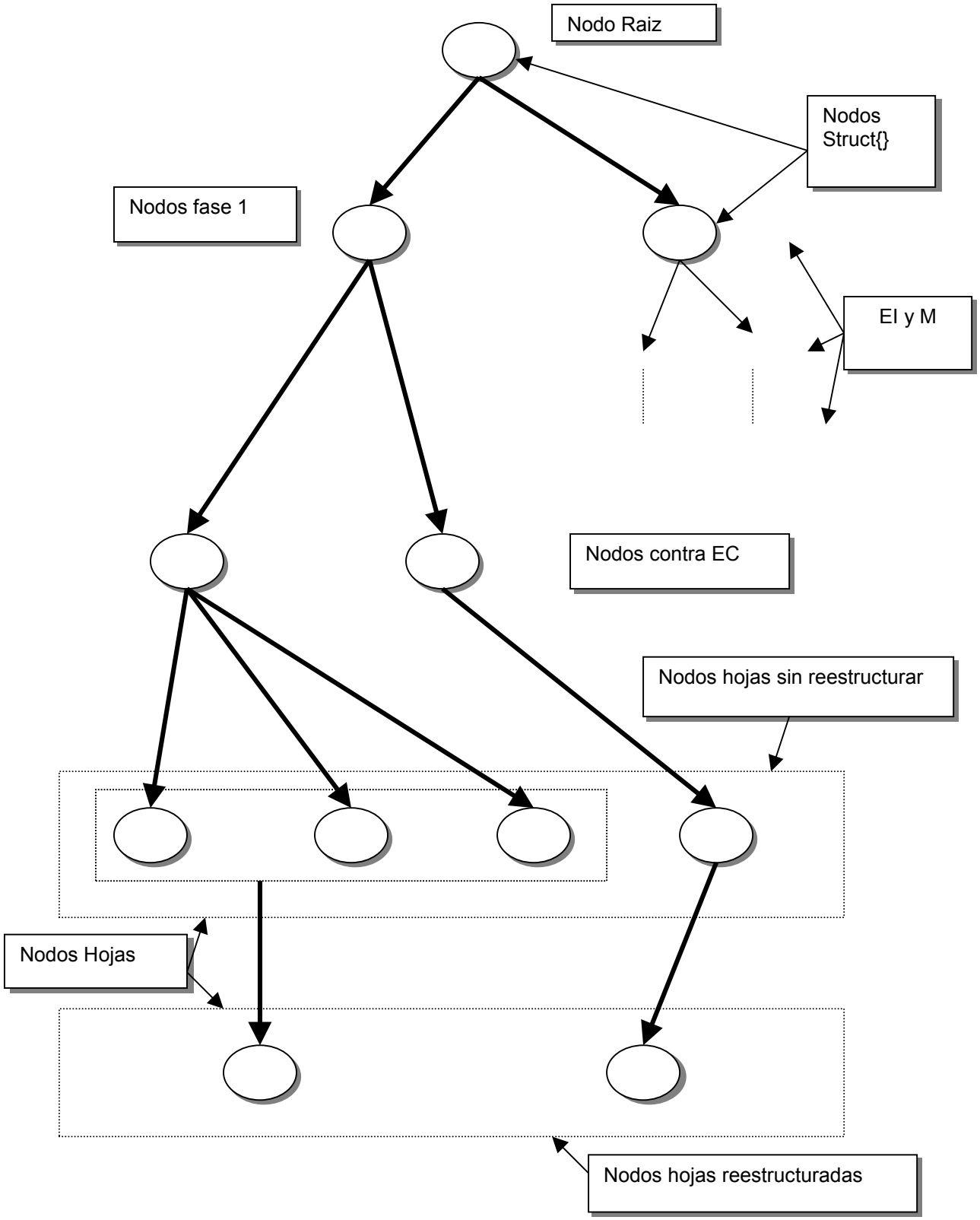
Si se observa que la clase virtual esta compuesta de varias clases, entonces la consulta es descompuesta en subconsultas, dependiendo de los ECs que haya, si hay 2, da lugar a 2 descomposiciones iguales, pero cabe señalar que la clausula From correlacionada a cada Esquema Componente ahora es diferente, ya que cuenta con un conjunto de clases en el intervalo de (c=1...n), donde "c" son las clases a materializar. Esta correlación permite dar a conocer las clases atómicas correlacionadas y por lo tanto su Esquema Componente. Esto ayuda a dividir la búsqueda y reorganizar las subconsultas en los nodos correspondientes con su funciones respectivas.

Una vez hecho lo anterior, el submódulo de descomposición procesa cada una de las subconsultas y detecta las clases atómicas que componen la clase virtual constituyendo un número de subconsultas iguales a las clases que forman la clase virtual. Es así como se crean los nodos que comprenden a estas subconsultas, para posteriormente considerar las cláusulas Where y Select con el objeto de tomar sus atributos derivados y procesar los predicados y en consecuencia depurar las consultas. Esta depuración involucra:

- Eliminar nodos
- Crear nuevos nodos
- Modificar subconsultas
- Analiza atributos
- Analiza *Path Expresión* (en predicados y proyección)
- Analiza Variables de Referencia
- Analiza Métodos de referencia
- Stand By de subconsultas

Se debe recordar que por la forma que el Esquema Integrado fue diseñado, los atributos discriminantes son de suma importancia para la correspondencias (mappings) y que debido a las dimensiones (Agregación, Especialización) del BLOOM las **aserciones** (equivalence assertion y specialization assertion) también son tomadas en cuenta sobretodo para la consolidación de resultados. Las "**path expression**" las cuales son producto de la operación de agregación es remplazada por su clase objeto. Se ejemplifica esta fase en la figura 5.

Función de Identificación de Objetos.- Otro punto importante son los oif, es decir, el procedimiento basado en la extensión, por el cual un objeto en una BDC denota el mismo objeto del mundo real en un objeto que pertenece a otra BDC. Si hacemos mención de [GSCP95], el cual abrevia que lo interesante no es la extensión de la bases de datos, sino sus denotaciones, también llamado su "estado del mundo real" (El estado del mundo real de una clase es el conjunto de objetos del mundo real representado por la extensión de las clases"), nos percatamos que esta función es de suma importancia. Para efectos de participación en el BLOOM se esta en investigación así que anexaremos al algoritmo de descomposición de la consulta un fragmento referente a los *oif*, suponiendo que están implementados y jugando su rol. Recuerde que un *oif* es una función Booleana con argumentos que son tomados de los atributos de la clase de la federación, aplicable a cada par de objetos de la clase federada.



EI: Esquema Integrado

M: Mapings

Figura 5. Ejemplo conceptual de la descomposición de una clase virtual contra los EC. Fase 2

Fase 3. Localizar Bases de Datos Componentes (sitios).

Después que el QD termine, la clase de cada subconsulta resultante debe ser localizada en una BDC única, y es entonces cuando las subconsultas son enviadas al módulo de traducción de la subconsulta (ver algoritmo 3), si la BDC mantiene un mismo lenguaje y un modelo que el sistema federado, la subconsulta pasa directamente a la BDC, de otra forma necesita del *Traductor de Consultas*.

========== Algoritmo 3 ==========

```
/* Este algoritmo encuentra sitios y etiqueta subconsultas */
Procedure Analiza_sitios (Nodo3[])
        ClaseSitio := get (Nodo3[], Clase)              /* toma de la subconsulta la clase
                                                           respectiva */
        While (ClaseSitio <> Diccionario.clases.mappings) do /* busca a que BDC hace
                                                           referencia esta clase */
                skip                                    /* continua la búsqueda
        End_while (Clases_Sitio)
        Nodo3[] := subconsulta + sitio                  /* Etiqueta la subconsulta, al
                                                           correspondiente nodo */
end_procedure

Procedure Envia (Nodo3[])
        ClaseEtiq := get (Nodo3[], Clase, Etiqueta)     /* toma de la subconsulta la clase y la
                                                           etiqueta respectiva y envía al traductor
                                                           de consultas */
end_procedure

Begin (Nodo3[], NN)                                     /* Toma de entrada las subconsultas y
                                                           el núm. de Subconsultas */
        Procedure Analiza_sitios (Nodo3[])              /* Toma atributos discriminados y con
                                                           el Diccionario apoya para encontrar
                                                           sitios */
        Procedure Envia (Nodo3[])                       /* Etiquetada la subconsulta, envía al
                                                           traductor de consultas, si es pertinente
                                                           */
end_Begin
```

========== Fin del Algoritmo 3 ==========

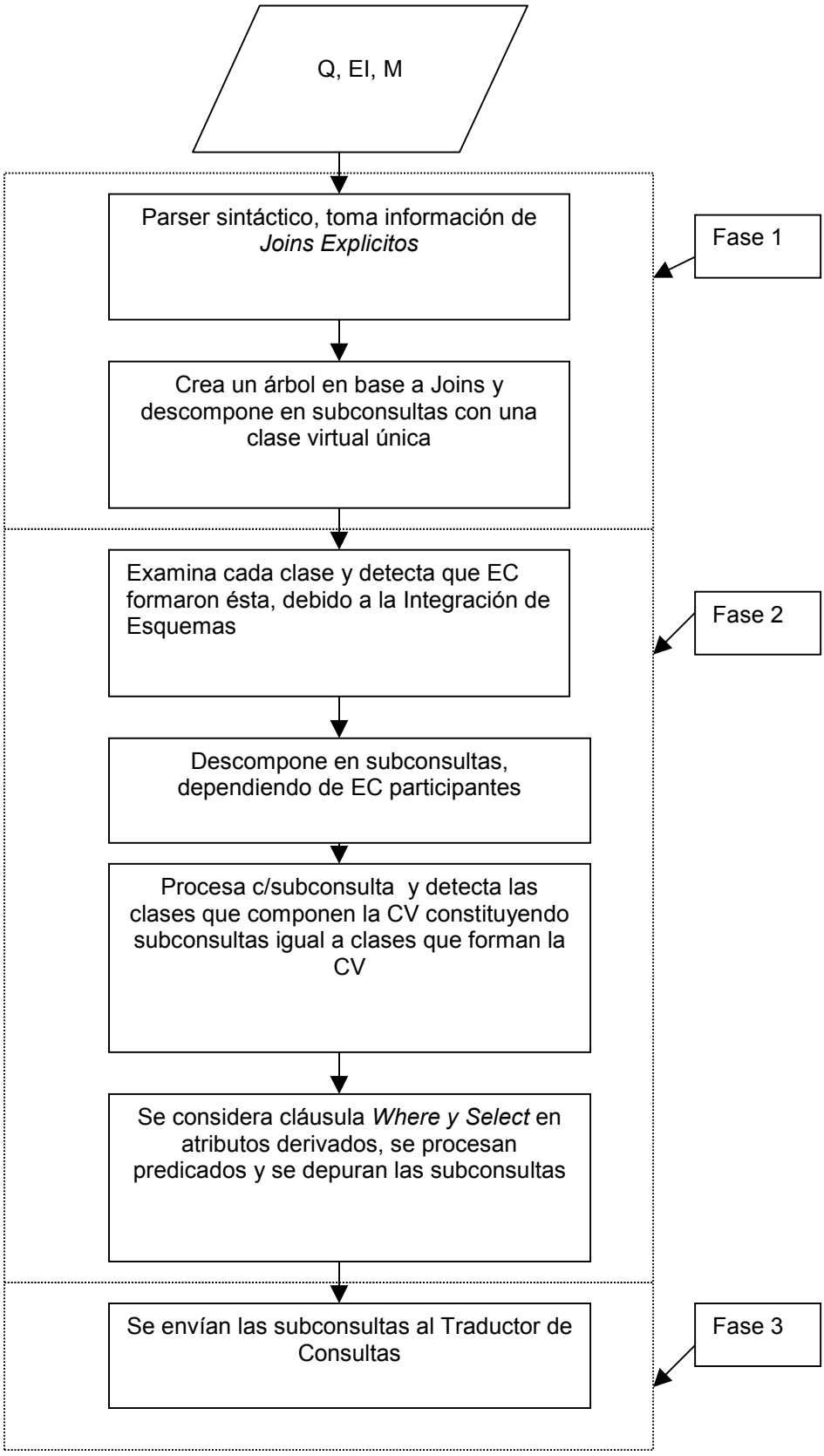


Fig. 6. Diagrama de flujo en bloques, del QD

5. Consolidación de Resultados

Consolidación de resultados. Una vez que las subconsultas son procesadas, se procede a regresar los resultados a la raíz, por lo que los subresultados son ahora ensamblados en el submódulo de Consolidación de Resultados, el cual pertenece al Gestor de Consultas. La tarea de este módulo será explicada y expresada en el algoritmo 4. Como es propuesto en [ROSC-P97], los subresultados son enviados al nodo padre (forma ascendente) al sitio de la consolidación de resultados. Este es apoyado por el módulo de transacciones. El árbol de ejecución de la consulta proyecta ésta consolidación, ver figura 7. Durante la consolidación los resultados son leídos con el conjunto de objetos de clases y sus respectivas correspondencias, considerando el Esquema Integrado para resolver las diferencias semánticas como lo resalta [CU-P96].

Funcion de gestión de Oids. Existe una función que gestiona los *Oids* locales y los *Oids* globales la cual encuentra la correspondencia *Oid* global de un objeto analizando una tabla de correspondencia *Oid global – Oid local* cuando un *Oid* es especificado, la técnica es incluir los *Oid* en las cláusulas Select para el posterior resultado de integración

========== Algoritmo 4 ==========

```
/* Este algoritmo es el encargado de realizar la integración de resultados, perteneciente al
submódulo Consolidación de Resultados dentro del Gestor de Consultas, para ello toma el
nodo restructurado (Nodo3) y sus correspondientes valores (set, bags de objetos) almacenados
en la estructura y apoyado de la función de gestón de Oids.*/
/* Declaración de variables
Struct Nodo3 [] {                   /* basado en la estructura de nodos, se toma información
                                       de sus correspondientes niveles */
        Padre := Nodo2[]
        Inf := Q
        Nivel := n
        Resp := set(objetos)
        Pos := n
}

Begin (Nodo3[], M, EI)              /* Nodos correspondientes, mappings, y esquema
                                       integrado */
  Procedure Get_NR (Nodo3[], NúmResp)   /* toma número de respuestas (sets)
                                       del total de subconsultas con clases atómicas*/

        for  (R_1 := 1, R_1<=NúmResp, R_1++) do        /* Ciclo de nodo hoja */
                seek Nodo3[] (Nodo3[].Pos == R_1)   /* Busca Nodo actual */
                Misma_clase := Nodo3[].Padre        /* mantiene la unión de clases de una
                                                       misma clase padre */
                InfTemp     := Nodo3[].Resp         /* Almacena respuesta de BDC en
                                                       variable temporal */
                seek Nodo3[] (Nodo3[].Pos == R_1+1)/* Busca Nodo sucesor */
                if (Nodo3[].Padre == Misma_clase)   /* pregunta si pertenece a la misma
                                                       clase padre */
                        then
                          Procedure Analiza_oif()    /* Identificación de objetos en nodos
                                                       hoja */
                          Procedure Analiza_null_values() /* valores nulos posibles que
                                                       afecten a la consolidación */
                          InfVirtual   := InfVirtual + OQL_union (InfTemp, Nodo3[].Resp)
                                                       /* Une todas las respuestas de una
                                                       subconsulta con clase virtual única */
                          InfTemp := conjunto vacio[]  /* Pone listo InfTemp */
                        Else
                          Seek Nodo2[] (Nodo2[].Pos == Misma_clase) /* Busca Nodo
                                                        Padre para almacenar respuesta */
```

```
                    Nodo2[].Resp := InfVirtual      /*Almacena Respuesta en Nodo
                                                    Padre */
                    NumResp := NumResp –1           /*Retrocede una posición del Nodo
                                                    para empezar a unir clases atómicas
                                                    de otra clase virtual */
          end_if (Misma_clase)
    end_for (NumResp)

    Procedure Get_NR (Nodo2[], Resp)                /* toma número de respuestas (sets)
                                                     del total de subconsultas */

    for  (R_1 := 1, R_1<=Resp, R_1++) do                    /* Ciclo de nodo intermedio */
          seek Nodo2[] (Nodo2[].Pos == R_1)         /* Busca Nodo actual */
          Misma_clase := Nodo2[].Padre              /* mantiene la unión de clases
                                                    de una misma clase padre */
          InfTemp     := Nodo2[].Resp               /* Almacena respuesta de
                                                    BDC en variable temporal */
          seek Nodo2[] (Nodo2[].Pos == R_1+1)/* Busca Nodo sucesor */
          if (Nodo2[].Padre == Misma_clase)         /* pregunta si es pertenece a
                                                            la misma clase padre */
                then
                  InfVirtual   := InfVirtual + OQL_union (InfTemp, Nodo2[].Resp)
                                                    /* Une todas las respuestas de una
                                                    subconsulta con clase virtual única */
                  InfTemp []:= conjunto vacio[]     /* Pone listo InfTemp */
                Else
                  Seek Nodo[] (Nodo[].Pos == Misma_clase) /* Busca Nodo
                                                     Padre para almacenar respuesta */
                  Nodo[].Resp := InfVirtual         /*Almacena Respuesta en Nodo
                                                                Padre */
                NumResp := NumResp –1               /*Retrocede una posición del Nodo
                                                    para empezar a unir clases atómicas
                                                    de otra clase virtual */
          end_if (Misma_clase)
    end_for (clases interemdias)
    Procedure Get_Resp_Nodo (Nodo[], Resp_Nodo) /* toma número de respuestas (sets)
                                                    del total de Esquemas Componentes */
          for  (R_1 := 1, Resp_Nodo, R_1++) do  /* Ciclo de nodos EC */
            seek Nodo[] (Nodo[].Pos == R_1)     /* Busca Nodo actual */

            InfTemp     := Nodo[].Resp          /* Almacena respuesta de
                                                BDC en variable temporal */
            seek Nodo[] (Nodo[].Pos == R_1+1)   /* Busca Nodo sucesor */
            InfVirtual   := InfVirtual + operador (InfTemp,
                              + Nodo[].Resp)    /* Respuestas seleccionadas
            InfTemp := conjunto vacio[]         /* Pone listo InfTemp */
          end_for
    NodoRaiz[] := InfVirtual                    /* pone respuesta única en raíz para
                                                enviar al siguiente submódulo.*/
```

end_Begin

========== Fin del Algoritmo 4 ==========

El algoritmo 4, propone integrar la respuesta global de tal forma que los nodos hojas mantengan las respuestas de las subconsultas y realizar un proceso de identificación de nodos por clase virtual y subir al nodo N_Final-1 con la información para cada clase virtual a ese nivel, esto quiere decir que un conjunto de objetos permanecen en espera para ser unidos en un nivel contra los Esquemas Componentes. Pero antes de este paso, los resultados son analizados para conocer si estos objetos son los mismos, directamente identificados por sus *oif*

(ya mencionados anteriormente) si es así, seguirán identificados hasta la respuesta final, y presentados como objetos idénticos.

El siguiente paso es la unión basado en sus esquemas componentes y la unión correspondiendo a las *clases virtuales* (Clases Federadas). Finalmente, el resultado se coloca en el nodo raíz, como lo muestra el algoritmo 4.

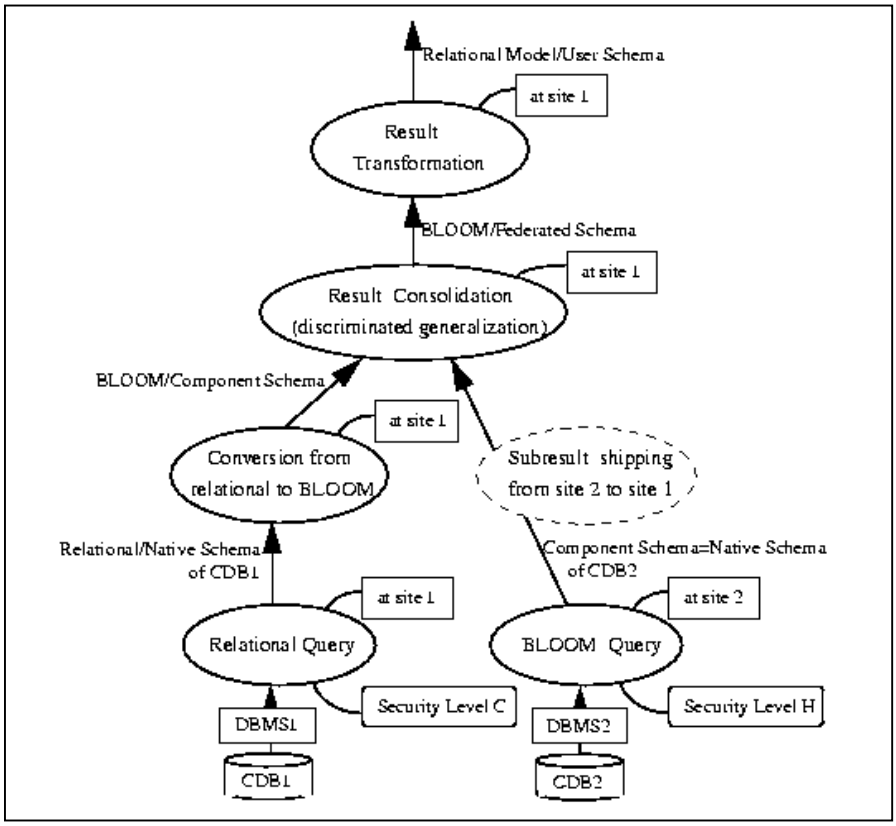


Figura 7. Árbol de ejecución de la consulta

6. Optimización de la Consulta Federada

Como se menciono en la introducción existen alguna técnicas empleadas para optimizar una consulta federada, sin embargo es un reto que sigue latente en las recientes investigaciones debido a los problemas que presenta la autonomía y heterogeneidad de las BDCs.

Generar excelentes planes de ejecución es un trabajo que empieza durante la descomposición global y las técnicas empleadas para realizar ésta. En la sección 4 se presenta una descomposición de la consulta global que amerita de una optimización en los nodos hojas reestructurados y es por lo tanto que una estrategia de predicados (ver algoritmo 5) se presentan en este documento para depurar y eliminar nodos innecesarios, al realizar esta técnica un árbol es formado, el cual permanece creado para ser procesado por el algoritmo 6, que optimiza la búsqueda por medio de los *path expression*, e interactúa con el Directorio del BLOOM. Estas heurísticas se mantienen para dar lugar al primer plan de ejecución después es presentado al módulo *Estimación del Costo* de la fase 3.

Para determinar el plan de ejecución optimo de la consulta global es necesaria información importante que llega de las BDCs y que en algunas ocasiones es imposible obtenerla y en otras solo parte de ella (parcialmente) debido a la autonomía de las BDCs. Esta es fundamental para el optimizador de la consulta global. Este problema puede ser solucionado coleccionando estadísticas sobre las BDCs por medio de técnicas como son expuestas en [WC-L95, BE-L96-Pegasus] o también técnicas de calibración como en [Du-P92, GST_P96], después de estas técnicas existe un mecanismo para evaluarlas y ellas son presentadas en [CDN_P92]. En este trabajo se mencionan como parte del algoritmo 5 para disponer de información de las BDCs Relacionales y Orientadas a Objetos que el modelo BLOOM dispone.

A continuación se presenta y propone el módulo de descomposición y optimización de la consulta global del Gestor de Consultas (ver figura 8). Para entender el proceso que se plantea, se ilustra un ejemplo que permite observar una integración de esquemas en BLOOM y con el cual planteamos una consulta que es utilizada en las secciones siguientes.

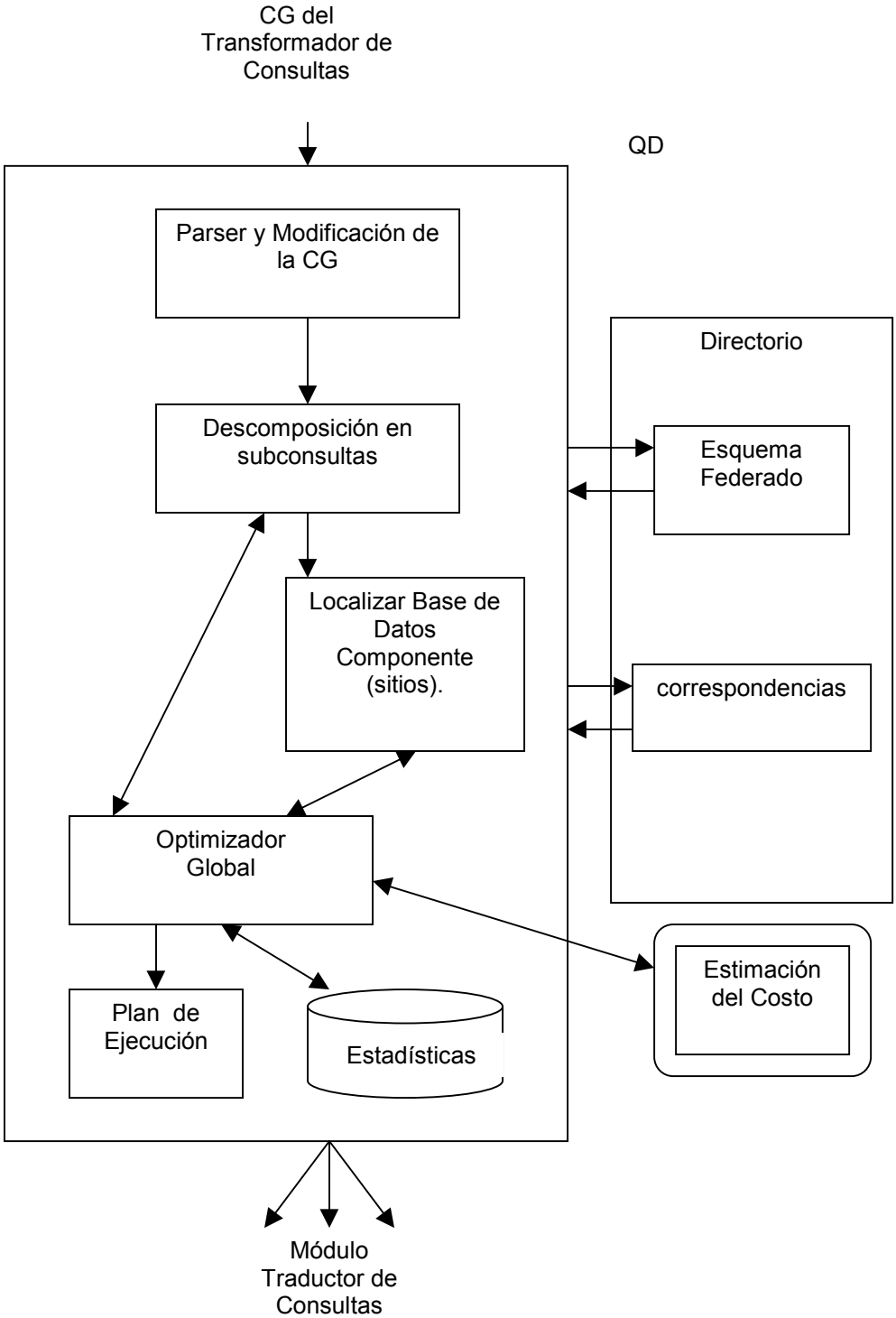


Fig. 8. Módulo de Descomposición y Optimización de Consultas Globales,
(Query Decomposer –QD-)

Para ejemplificar el proceso de optimización de la consulta a nivel federado se presenta la integración de un esquema global sencillo, que se utiliza en el proceso de la explicación de este documento. Este toma en cuenta el Esquema Integrado y los Mappings del BLOOM. A continuación se presenta la figura 9a) que muestra las clases para esquematizar conceptualmente la Base de Datos 1 (BD1) en la figura 9b).

```
Class Estudiante {
        disj_gen_of E_Posgrado, E_Licenciatura [by nivel]
        simple_aggreg_of
                carnet: int;
                nombre: strings obligatory;
                edad: int;
                univ: strings;
                dirección: Dirección;
                vehículo: Vehiculo;
                …
                id carnet;
}
end_class
```

```
class E_Posgrado {
        disj_spec_of Estudiante
        simple_aggreg_of
        ...
}
end_class
```

```
class E_Licenciatura {
        disj_spec_of Estudiante
        simple_aggreg_of
        ...
}
end_class
```

```
class Dirección {
        simple_aggreg_of
                calle: strings
                num: int;
                ciudad: strings;
                ...
}
end_class
```

```
class Vehículo {
        simple_aggreg_of
                mat: int;
                modelo: int;
                marca: strings;
                ...
}
end_class
```

Figura 9a

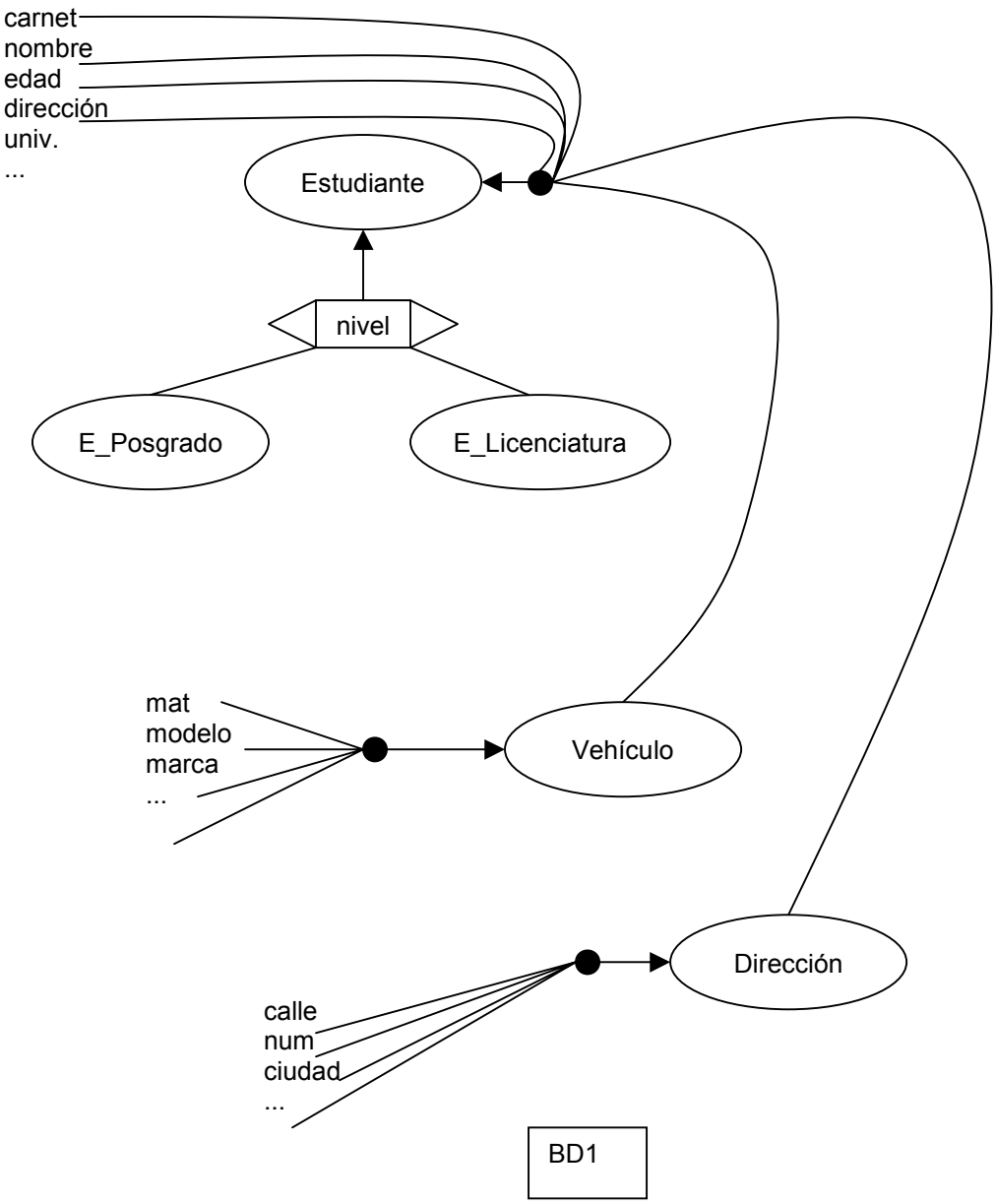

carnet
nombre
edad
dirección
univ.
...
Estudiante
nivel
E_Posgrado
E_Licenciatura
mat
modelo
marca
...
Vehículo
Dirección
calle
num
ciudad
...
BD1

Figura 9b

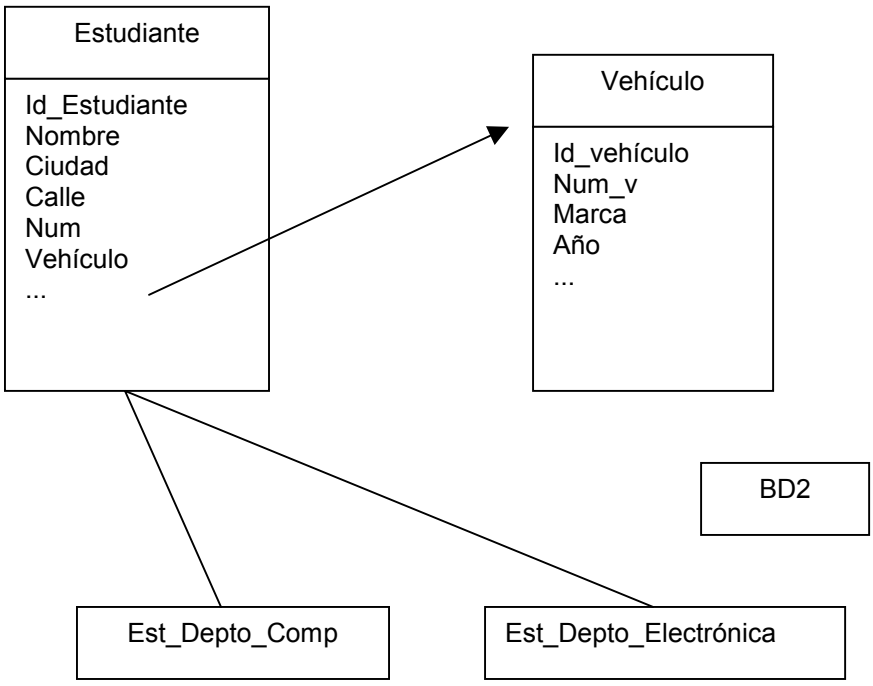


Figura 9c

La figura 9c) es la representación del Esquema Conceptual con sus relaciones de la Base de Datos 2 (BD2). Esta BD2 se encuentra en un sitio diferente a la BD1. La BD1 y la BD2 son dos universidades diferentes. En estas bases de datos se almacena la información perteneciente a los alumnos. Este ejemplo no pretende visualizar los detalles, solo se utiliza para ejemplificar una **descomposición y optimización de consultas** a nivel federado dentro del Gestor de Consultas. Una vez que el Esquema Integrado queda terminado y sus mappings cubiertos, llega la fase de ejecución, en la que una Arquitectura de Operación entra en proceso. Para ello contamos con el Esquema Federado mostrado en la figura 9e), el cual se integro por los criterios tomados de [AORS99, AORS99-2, CORRS-96, GSCP93, SCGP93 y ROSC-P97], también presentamos sus clases virtuales en la figura 9d).

```
Class G_Estudiante {


 disj_gen_of G_Posgrado, G_Licenciatura
                                    [by nivel]
 alt_gen_of Estudiante_bd1, Estudiante_bd2
                                    [by bd_discr]
        simple_aggreg_of
                bd_discr: [bd1, bd2]
                G_num: int;
                G_nombre: strings;
                G_edad: int;
                G_depto: int;
                G_univ: strings;
                G_dirección: G_Dirección;
                G_vehículo: G_Vehículo;
                ...
                id G_num, bd_discr;
}
end_class
```

```
class G_Posgrado {
        disj_spec_of G_Estudiante
        simple_aggreg_of
        ...
}
end_class
```

```
class G_Licenciatura {
        disj_spec_of G_Estudiante
        simple_aggreg_of
        ...
}
end_class
```

```
class G_Dirección {
        alt_gen_of G_dir_bd1, G_dir_bd2 [by
bd_discr]
        simple_aggreg_of
                bd_discr: [bd1, bd2];
                G_ciudad: strings;
                G_calle: strings;
                G_no: int;
                G_cp: int;
                …
                id bd_discr
}
end_class
```

```
class G_Vehículo {
        alt_gen_of G_veh_bd1,
G_veh_bd2 [by bd_discr]
        simple_aggreg_of
                bd_discr: [bd1, bd2];
                G_marca: strings;
                G_año: int;
                G_matrícula: int;
                …
                id bd_discr
}
end_class
```

Figura 9d

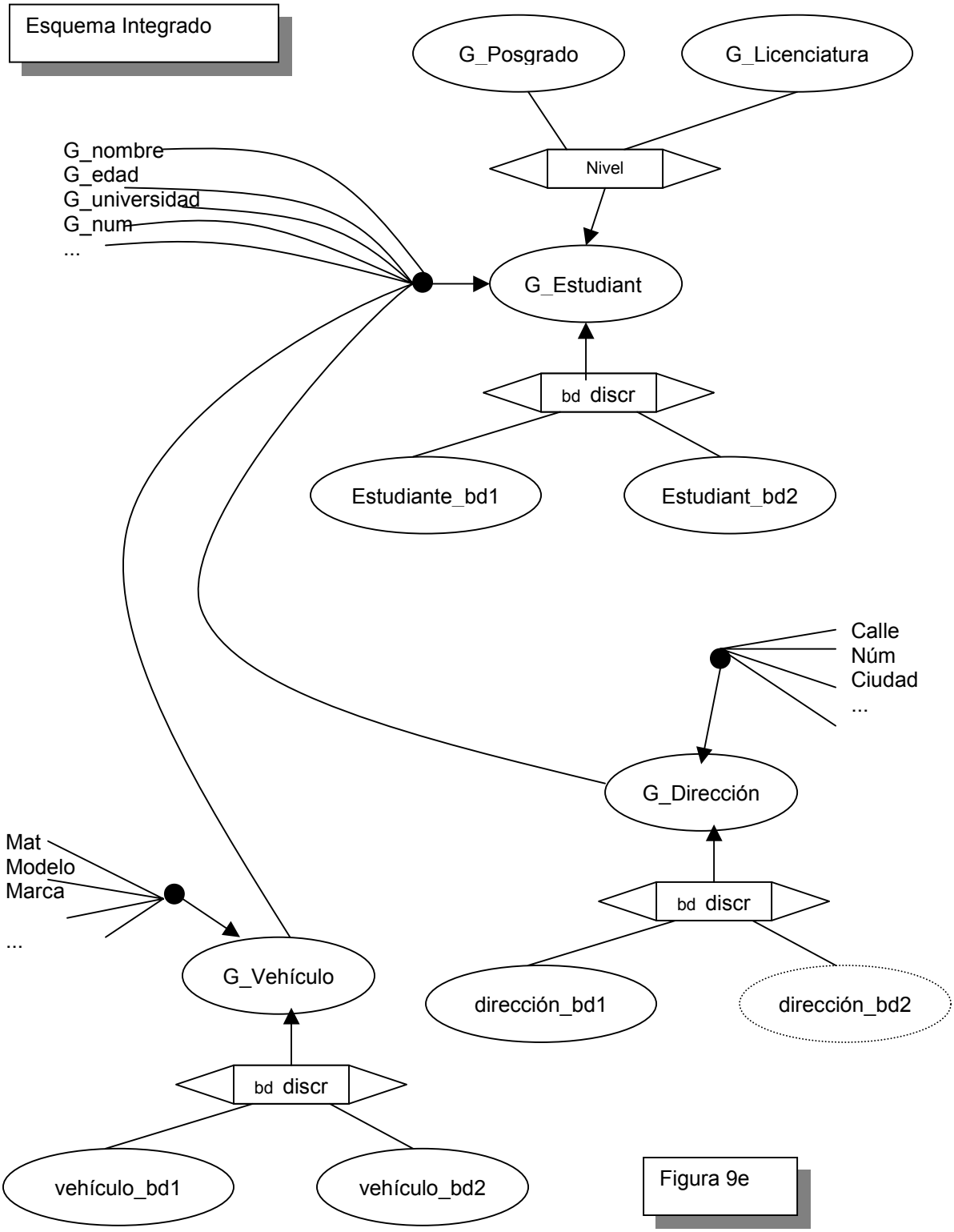

Esquema Integrado
G_Posgrado
G_Licenciatura
Nivel
G_nombre
G_edad
G_universidad
G_num
...
G_Estudiant
bd discr
Estudiante_bd1
Estudiant_bd2
Calle
Núm
Ciudad
...
G_Dirección
Mat
Modelo
Marca
...
G_Vehículo
bd discr
dirección_bd1
dirección_bd2
bd discr
vehículo_bd1
vehículo_bd2

Figura 9e

6.1. Técnicas heurísticas para optimizar la fase 2

Antes de iniciar la explicación de la optimización para la fase 2, debe quedar claro que la fase 1 correspondiente a la ejecución de un Parser permanece tal y como esta, debido a lo propuesto en [CORRS-96]. El lenguaje que se está desarrollando es de tipo declarativo y muchas de sus instrucciones tienen la forma de nuevas modalidades y variantes de la operación de agregación, con lo que aparece como una extensión del modelo BLOOM más que como un lenguaje simplemente basado en dicho modelo; muchas de estas instrucciones pueden ser utilizadas tanto para definir aspectos permanentes de la BD como para la generación de resultados de consultas. La especificación de la sintaxis y la semántica del lenguaje se harán mediante técnicas formales. La implementación tendrá lugar sobre SGBDs OO y Relacionales. Y como se puede observar en la figura 4 de la sección 4, la modificación de la consulta federada que se aplica no necesita una optimización compleja, ya que dentro del procesamiento de la misma existe un algoritmo llamado Modi_Con(x_1,x_2,...x_n) donde los x_i son los parámetros de entrada y que aplican procesos de ordenación aparte de su modificación establecida.

El algoritmo 2 toma una Consulta Global Modificada de la fase 1 (ver figura 5), y realiza una descomposición en subconsultas, la cual procesa una consulta con a lo más una clase virtual global y que a su vez pueda contener varias clases en diferentes bases de datos componentes, resultado de una operación de integración (en BLOOM, las dimensiones de agregación y generalización son base de su integración [CORRS-96]). Este algoritmo se corresponde con la fase 2 y para efectos de optimización de la consulta, en esta sección nos centramos en los siguientes puntos:

1) Eliminar nodos
2) Crear nuevos nodos
3) Modificar subconsultas
4) Analizar atributos
5) Analizar Path Expression (PE)
6) Analizar variables de referencia
7) Analizar métodos de referencia

6.1.1. Procesamiento y Optimización a través de un ejemplo propuesto

Para observar las estrategias de optimización presentamos un ejemplo, el cual es una consulta federada hecha por un *usuario u*, en un *sitio s*:

Ejemplo 1. Obtenga el nombre y la universidad de los alumnos de postgrado y licenciatura que radiquen en la ciudad de Tarragona, que además sean menores de 25 y mayores de 20 y que finalmente tengan un vehículo de la marca “Renault” (suponiendo que la mayoría de los alumnos tienen vehículo). La expresión puede ser presentada en OQL como sigue:

Consulta 1

```
Select  gs.nombre, gs.universidad
From G_Estudiante gs
Where gs.nivel = Posgrado or gs.nivel = Licenciatura
        and gs.edad<25 and gs.edad>20
        and gs.Dirección.ciudad = “Tarragona”
        and gs.Vehículo.marca = “Renault”
```

El flujo del procesamiento implica pasar por el primer submódulo del Gestor de Consultas llamado “Query Transformer” (este documento no abarca este submódulo) y posteriormente entramos a la primera fase del “Query Decomposer” (QD) el cual descompone la consulta federada en subconsultas contra esquemas componentes. En el caso del ejemplo particular la fase 1 solo aplica el Parser, debido al no procesamiento de Joins Explicitos, observe solo la aparición de una CLASE VIRTUAL G_Estudiante.

La fase 2 entra en proceso y hace un análisis de sus Joins Implícitos y detecta que la clase virtual esta construida de la BDC1 y BDC2, por lo que la subconsulta 2 y 3 deben ser construidas según el proceso de descomposición, ver subconsultas a continuación:

Subconsulta 2
Select gs.nombre, gs.universidad
From *Clase_Virtual_BD1* /* Contra Esquema Comp.
Where gs.nivel = Posgrado or gs.nivel = Licenciatura
and gs.edad<25 and gs.edad>20
and gs.Dirección.ciudad = “Tarragona”
and gs.Vehículo.marca = “Renault”

Subconsulta 3
Select gs.nombre, gs.universidad
From *Clase_Virtual_BD2* /* Contra Esquema Comp.
Where gs.nivel = Posgrado or gs.nivel = Licenciatura
and gs.edad<25 and gs.edad>20
and gs.Dirección.ciudad = “Tarragona”
and gs.Vehículo.marca = “Renault”

Hasta el momento parte del árbol de descomposición se desarrolla como lo muestra la figura 5, y donde empieza el proceso de la optimización, ya que después de construir los nodos intermedios contra esquemas componentes se inicia la construcción de los nodos hojas, los cuales mantienen solo clases atómicas y que son optimizadas por el Algoritmo 2. Tres son las subconsultas (4, 5 y 6) que corresponden a la BDC1, y esto se debe a las correspondencias que permanecen en el DIRECTORIO después de la Integración de Esquemas. Recuerde Fase de Detección y Fase de Resolución del BLOOM. A continuación las subconsultas 4, 5 y 6:

Subconsulta 4
Select gs.nombre, gs.universidad
From *Estudiante%BD1 gs*
Where gs.nivel = Posgrado or gs.nivel = Licenciatura
and gs.edad<25 and gs.edad>20
and gs.Dirección.ciudad = “Tarragona”
and gs.Vehículo.marca = “Renault”

Subconsulta 5
Select gs.nombre, gs.universidad
From *E_Posgrado%BD1 gs*
Where gs.nivel = Posgrado or gs.nivel = Licenciatura
and gs.edad<25 and gs.edad>20
and gs.Dirección.ciudad = “Tarragona”
and gs.Vehículo.marca = “Renault”

Subconsulta 6
Select gs.nombre, gs.universidad
From *E_Licenciatura%BD1* gs
Where gs.nivel = Posgrado or gs.nivel = Licenciatura
and gs.edad<25 and gs.edad>20
and gs.Dirección.ciudad = “Tarragona”
and gs.Vehículo.marca = “Renault”

Ahora tenemos los nodos hojas finales que son procesados por el algoritmo de la fase 2 en paralelo con el submódulo de optimización (ver figura 8, optimizador global), y siguiendo los pasos del Algoritmo 5, iniciamos con el Análisis de Predicados. Es decir; se aplican procesos para crear un árbol de predicados, el cual lleva a cabo operaciones sobre las proposiciones presentadas por la cláusula WHERE. Tomando en cuenta operadores binarios, se realiza un proceso sobre **atributos derivados** que pertenezcan a esa clase y elimina subconsultas si no hacen mención a ella, para este ejemplo solo uno es eliminado y es la subconsulta con la clase

Estudiante (ver figura 10). Recuerde que también tenemos la subconsulta 2 que sigue correspondiendo a la BDC2 y que llamaremos ahora la subconsulta 7, presentada como sigue:

Subconsulta 7
Select gs.nombre, gs.universidad
From *Estudiante gs*
Where gs.nivel = Posgrado or gs.nivel = Licenciatura
and gs.edad<25 and gs.edad>20
and gs.Dirección.ciudad = “Tarragona”
and gs.Vehículo.marca = “Renault”

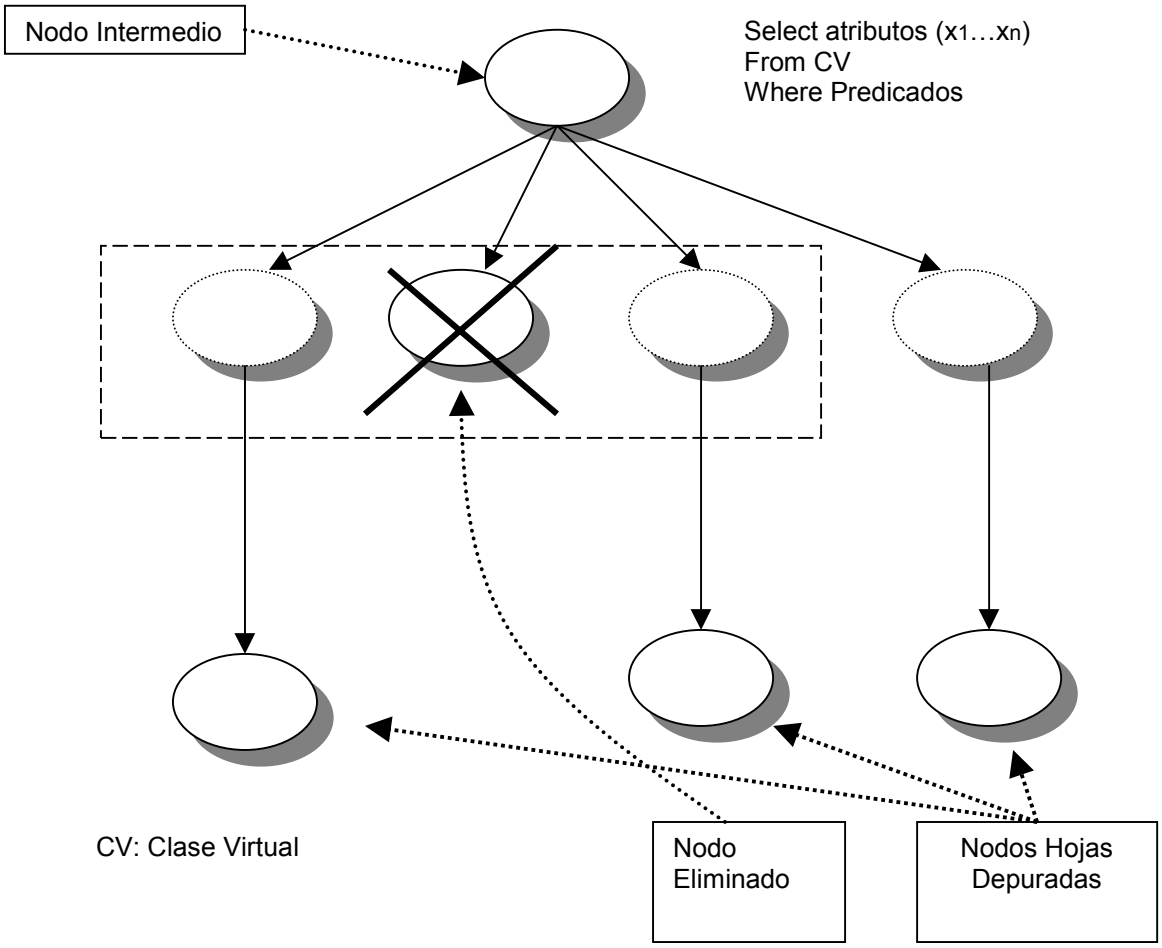


Figura 10. árbol de subconsultas después de análisis de predicados

Una vez creado el árbol de predicados pasamos al algoritmo de **path expression**, muy importante y que mantiene la complejidad del proceso de optimización buscando el mejor **Plan de Ejecución**, vea el Algoritmo 6.

========== Algoritmo 5 ==========
/* Analiza_Predicados()
/* En este algoritmo (procedure del algoritmo 2), se obtienen primeramente los predicados de las hojas (predicados reestructurados) y genera un árbol de predicados para determinar si cada subconsulta es factible para enviar a la BDC correspondiente, a través de mapeo y siguiendo la Regla 1.*/

Procedure Analiza_Predicados(Nodo3[])

```
get(Nodo3[].Inf, Q[j])                          /*Inicia recuperación de Subconsulta del
                                                nodo hoja sin reestructurar. */
get(Q[j], ClausulaW)                            /*Mantiene Predicados {p1,...,pn}
                                                correspondiente a la clausula Where */
Procedure EjecCompilaTree(ClausulaW,Pw,T_Pred) /*Aplicando una compilación
                                                simple, genera el número total de predicados
                                                en la clausula, y genera un árbol de
                                                predicados Pw*/
for (i:=1,i<=T_Pred,i++) do                     /*Ciclo para aplicar Regla 1*/
        F_Valida_correspondencia(Pw[Nodo3[].pos:=i]) /*La Función regresa un valor
                                                bolean, después de examinar correspondencia
                                                de atributos derivados contra Esquemas
                                                Componentes. Si es True mayor
                                                correspondencia. */
```

/***Regla 1**. Dentro de un conjunto de nodos hojas set(NH), con la misma clase virtual única Ψ contra el EC, se debe llevar a cabo un análisis exhaustivo de las correspondencias con las clases de los esquemas de las BDCs presentadas para la integración, apoyada por el Directorio. Esto es para encontrar los NH que sean factibles de procesar con esos predicados de la subconsulta.*/

```
        end_for(T_Pred)
```

end_Procedure

Porcedure EjecCompilaTree(ClausulaW,Pw,T_Pred) /*Crea un árbol binario para los Predicados */

```
scan(Pw)                                        /*lee el primer predicado y lo inserta
                                                en un árbol binario de nodo único */
tree[]:=P_MakeTree(Pw);                         /* hace primer recorrido creando
                                                una ruta hasta atributo derivado */
while (mientras haya predicados){
                scan(Pw); .....                 /* toma siguiente predicado */
        if Pw == NodoRaiz                       /*pregunta si es PE*/
                then
                        NodoRaiz.inf:=Pw        /*deja información en nodo raíz ya que
                                                no es una PE*/
                else
                        P_Der_Izq(Pw)           /*sigue un camino de derecha o
                                                izquierda, hasta encontrar su hoja */
        End_if
}/fin while
```

end_Procedure

========== Fin del Algoritmo 5 ==========

6.1.2 Path Expresión

Path expression es un mecanismo simple pero poderoso para especificar navegación a lo largo de una composición jerárquica en un esquema orientado a objetos. Estas aparecen frecuentemente en las clausulas Where, aunque también en las cláusulas *Select* y *From*, pero para este trabajo aplicaremos solo a cláusulas *Where*.

La optimización de *path expression* han recibido una atención importante porque son construcciones fundamentales para especificar consultas en bases de datos orientadas a objetos y los SGBDF no se quedan atrás. Es posible usar álgebra de objetos para generar

diferentes planes de ejecución para procesar *path expression*, en [YW-L97] se propone analizar el problema en un nivel intuitivo y asumiendo que los *path expression* son single-valued, es decir que no tienen un conjunto de atributos en las *path expression* y solo se proponen para las BDCs que mantenga un esquema orientado a objetos. Para la BDC que su esquema es otro ej: Modelo Relacional, se sustituyen las *path expression* por sus correspondientes clases con la ayuda del Directorio Global.

Definición: Para una consulta dada, se define su grafo de consulta como un grafo dirigido: GO(VO,AO), donde VO es un conjunto de vértices y AO es un conjunto de arcos dirigidos. Cada vértice *v* en VO es asociado con dos tipos de expresiones, de predicado y de objetivo. El primero contiene todos lo predicados en la correspondiente variable de instancia de clase, el último contiene todos los atributos finales de la correspondiente variable de instancia de clase. Un arco dirigido *a* de un vértice *v1* a un vértice *v2* con expresión *Attr* indica que *Attr* es un atributo complejo de la clase correspondiente a *v1* y la clase dominio de Attr corresponde a *v2*.

Se necesitan métodos heurísticos para guiar el proceso de seleccionar una buena estrategia de ejecución. Se presenta una heurística basada en bloques (block-based), este método consiste de dos pasos. En el primer paso, se genera un grafo de consulta y es transformado en un árbol con bloques, donde cada subárbol es algún vértice interno del árbol original (ver figura de Ejemplo 2). En el segundo paso, las subconsultas que corresponden a bloques son evaluadas de una manera dentro-fuera; es decir, que las subconsultas que están en los bloques mas internos, son evaluadas primero. Cuando un bloque es evaluado, la raíz del correspondiente subárbol es reducido; es decir, el conjunto de instancias de la clase correspondiente a la raíz del subárbol que satisface las condiciones impuestas en el subárbol son retornadas como un resultado intermedio y que a nivel federado éste resultado intermedio será el **atributo materializado**. Conceptualmente después de que un bloque es evaluado, este es reducido a un vértice único correspondiente a la raíz del subárbol, la siguiente definición aplica a su configuración:
Definición: Para un árbol de consulta, con *path expression*, una configuración de un bloque es un conjunto de bloques que satisfacen las siguientes condiciones:

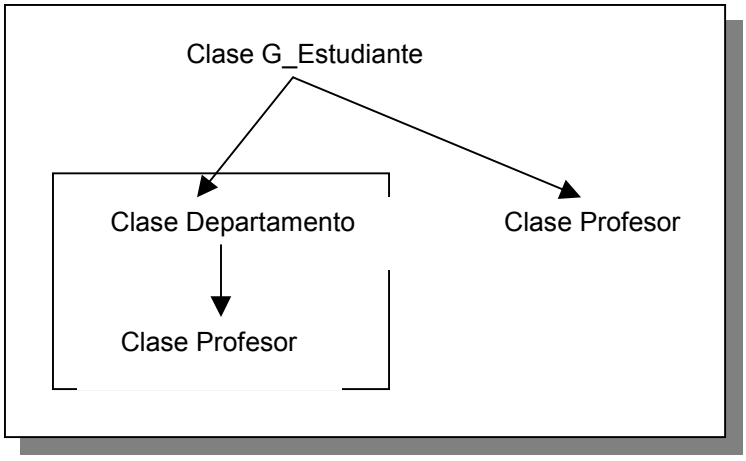
1. Cada vértice de un árbol esta en al menos un bloque.
2. Cada bloque contiene al menos dos vértices.

Después de conocer su metodología, lo importante es encontrar una configuración de bloque que mejore el plan de ejecución, ya que un sinnúmero de configuraciones pueden ser encontradas, recuerde que solo aplica esta metodología a la BDC que es OO. A continuación se presenta un ejemplo que se evalúa por sus atributos complejos:

Ejemplo 2. Encuentre todos los estudiantes graduados que están en un departamento cuyo titular es el Dr. Felix Saltor y que tengan un tutor de tiempo completo menor de 40 años. La consulta se expresa en OQL como sigue:

```
Select s
From G_Estudiante s
Where s.Departamento.Titular.Nombre = Felix Saltor
        and s.Tutor.Tipo_contrato = “Tiempo Completo”
        and s.Tutor.Edad < 40
```

y cuyo grafo de consulta en bloques sea:



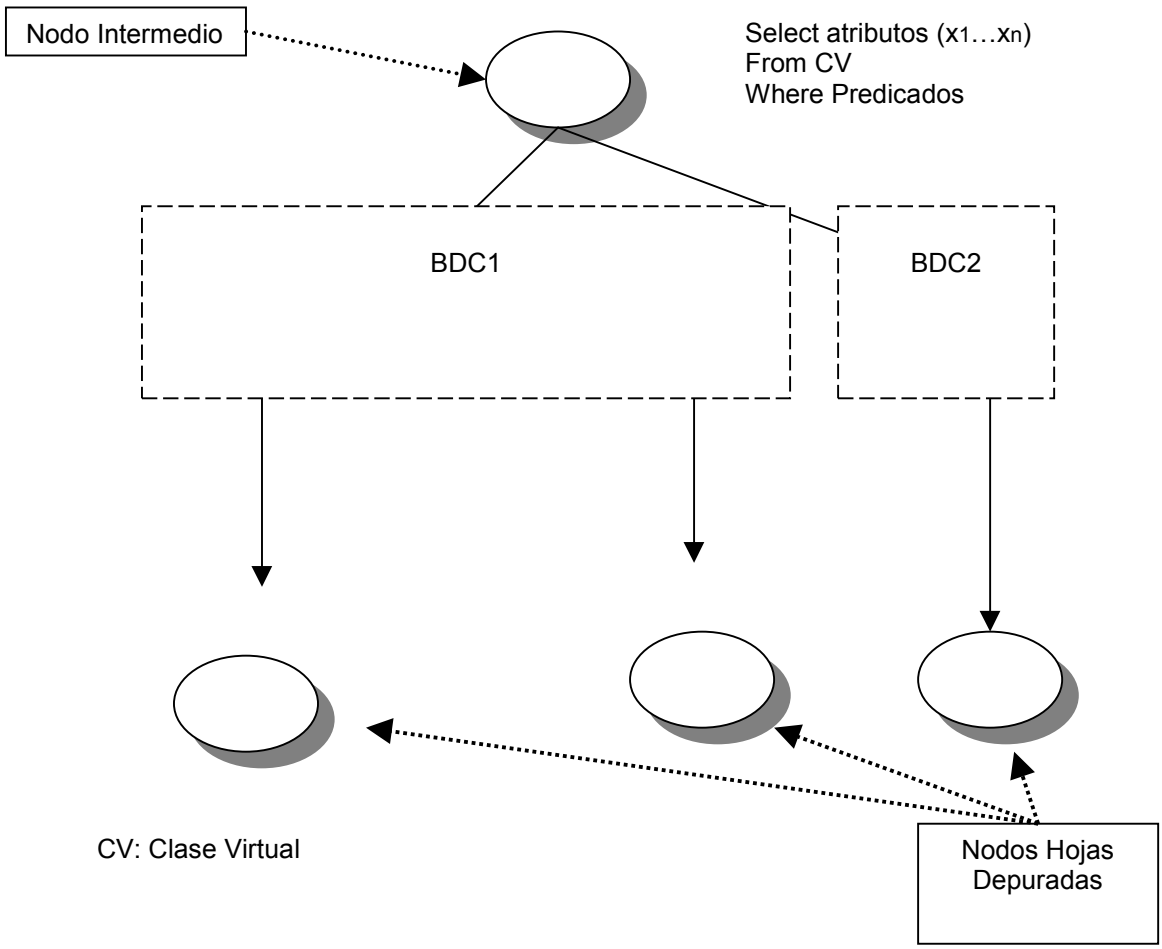


Figura 11. Representación de nodos hojas iniciando proceso de Plan de Ejecución por medio de Path Expresión, para el ejemplo en particular.

Para seguir con el ejemplo 1, se presenta en la figura 11 los nodos hojas depuradas, en la cual se aplica una técnica heurística basada en bloques para tener diferentes planes de ejecución y buscar el mejor de ellos, el cual será tratado en la siguiente sección, por ahora se presentan las subconsultas después del *PROCEDURE PATH_EXPRESIÓN*:

Subconsulta 8
Select gs.nombre, gs.universidad
From *E_Posgrado%BD1 gs*
Where and gs.edad<25 and gs.edad>20
and gs.Dirección.ciudad = “Tarragona”
and gs.Vehículo.marca = “Renault”

Subconsulta 9
Select gs.nombre, gs.universidad
From *E_Licenciatura%BD1* gs
Where and gs.edad<25 and gs.edad>20

```
and gs.Dirección.ciudad = “Tarragona”
      and gs.Vehículo.marca = “Renault”
```

Subconsulta 10

```
Select  nombre, universidad
From Estudiante%BD2, Vehículo%BD2
Where  and edad<25 and edad>20
       and ciudad = “Tarragona”
       and marca = “Renault”
```

Como observamos **atributos complejos** son encontrados en la subconsulta 8 y 9, por lo que el proceso se aplica para encontrar el mejor plan de ejecución. Un grafo dirigido es creado internamente para su análisis, pero como es un ejemplo sencillo que no crea bloques, solo tiene 2 niveles, no amerita representarlo.

El ejemplo anterior muestra como las *path expression* siguen un camino hasta encontrar un atributo derivado, y cuando este es alcanzado se sustituye por correspondiente en BDC ayudado por correspondencias ya establecidas. A continuación se presenta su algoritmo.

========== Algoritmo 6 ==========

```
/*Analiza_path_expression

Procedure Analiza_path_expression()
        While (para todos los Nodos Nodo3[]) Do      /* Toma todos los nodos hojas y sus
                                                     correspondientes árboles de
                                                     predicados, para evaluarlos */
                Seek(Nodo3[].Pos, Tree_Predicados)   /* busca y almacena en memoria */
                NTN = ∑(posibles nodos a evaluar);   /* NTN Numero Total de Nodos */
        End_while
        if PrimeraEvaluación
                Procedure Construye_Plan_Ejecución(N_Plan)  /* toma la primera forma de
                                                     construcción como Plan de Ejecución
                                                     1*/
                PlanEjecucion[N_Plan] = Nodo3[],Tree_Predicados;      /* Almacena en BD
                                                     Estadísticas primer Plan de ejecución
                                                     a evaluar*/
        End_if
        Procedure Aplica_Heuristicas_Bloqueo         /*realiza una serie de técnicas
                                                     heurísticas basadas en el bloqueo,
                                                     para escoger una configuración ver
                                                     [YW_L97] */

        REPEAT
        For (i=1;i<=NTN;i++) do
                Get(Nodo3[].Pos:=i,NodoFactible);
                if NodoFactibe
                        Then
                                NodosEvaluar+=1;
                                Procedure Construye_Plan_Ejecución(N_Plan)  /* toma la
                                                     siguiente forma de construcción como
                                                     Plan de Ejecución n+1*/
                        End_if
                                PlanEjecucion[N_Plan] = Nodo3[],Tree_Predicados;
                                                     /* Almacena en BD Estadísticas Plan
                                                     de ejecución a evaluar */
```

End_for(NTN)
UNTIL todas las configuraciones
end_Procedure

========== Fin Algoritmo 6 ==========

Representando y continuando con el ejemplo 1, la posible ruta aplicando el algoritmo 6, puede quedar como sigue después de tomar en cuenta la técnica heurística citada anteriormente.

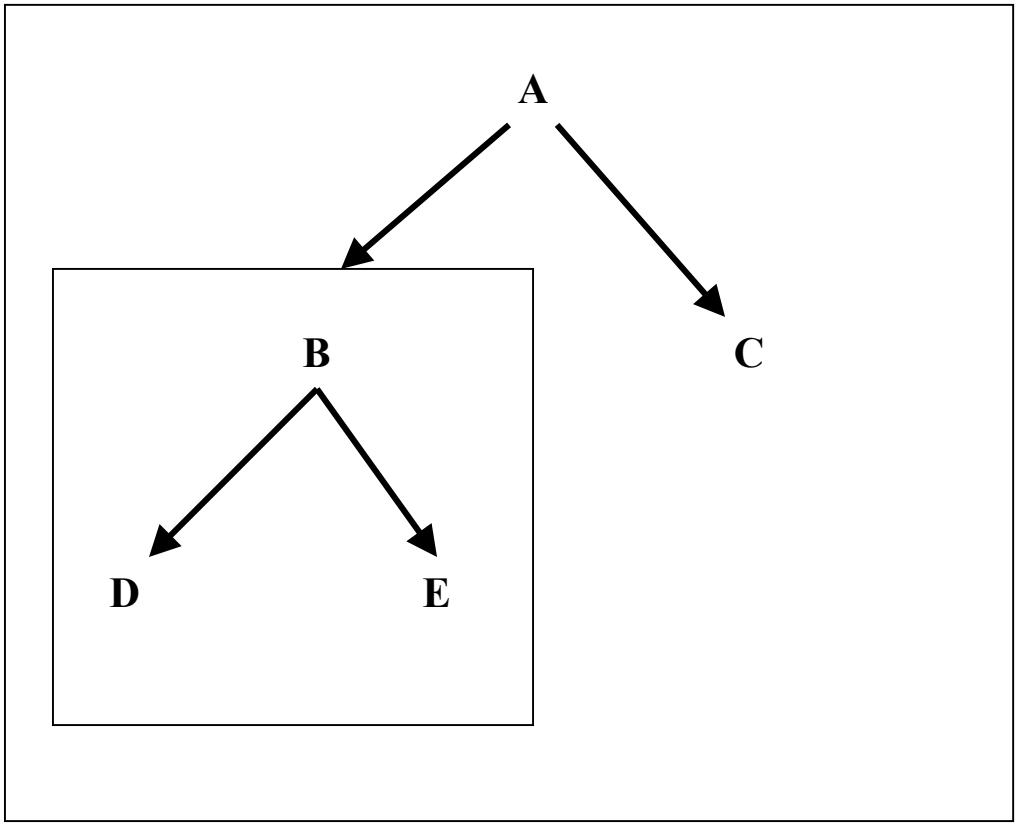


Figura 12. Representando Técnicas Heurísticas de Optimización (PE) para ejemplo 1.

Observando la Figura 12, debemos de tener en mente la participación de las correspondencias entre las clases virtuales y las clases en los ECs, ya que por medio de éstas las técnicas heurísticas son llevadas a cabo, en esta figura el nodo A, ilustra el nodo padre empezando en la segunda fase del submódulo de optimización, y cuando el algoritmo 5 y 6 son procesados, un plan de ejecución es creado para ser evaluado posteriormente. Este Plan de entre varios, es generado por medio de las combinaciones preestablecidas que las heurísticas presentan.

Un Plan de Ejecución para el ejemplo 1, es enviar a las BDCs los nodos D y E, y mantenerlos para que el **Módulo de Consolidación de Resultados** los procese junto con el nodo C, esta es solo un posible Plan de Ejecución de los que ya se tienen en su BD para Estadísticas. **Técnicas de paralelismo** son aplicadas (explicadas en otra sección de este trabajo después de que este PE, fue seleccionado como el optimo.

6.2. Plan de Ejecución de la Consulta en Fase 3

La optimización de la consulta es uno de los mayores retos dentro de los sistema de bases de datos federados.

En este trabajo se propone una optimización de 3 fases, en la primera fase vista en el punto 6.1, las consultas son simplificadas y procesadas usando algunas técnicas heurísticas. Como se pudo observar, la propuesta es obtener nodos hojas que sean reestructurados y que

simplifiquen la difícil tarea de evaluar y optimizar una consulta para obtener el mejor Plan de Ejecución.

El segundo paso en la optimización y que corresponde a esta sección es encontrar un Plan de Ejecución Optimo apoyándonos en una buena estimación de costos de ejecución. La propuesta que se plantea es encontrar un modelo de costos y un proceso de calibración desarrollado principalmente para las BDCs, es decir, la ejecución de unas técnicas que hagan posible estimar el costo de las consultas federadas, aunque la información de las BDCs sea mínima.

El objetivo principal del segundo paso de la fase 2 es generar un plan de ejecución en términos del uso de recurso totales, sin embargo como se puede observar este plan de ejecución no toma en cuenta el tiempo, solo los recursos, por lo tanto, se propone también una segunda optimización con respecto al tiempo. **El paralelismo** será un punto importante para resolver el menor tiempo posible.

Tendremos en mente la siguiente regla, para apoyarnos en la optimización del tiempo:

Regla 2. Si el objeto de la clase A y el objeto de la clase B están almacenados en el mismo *sitio i*, entonces el Costo Total de recuperar el objeto de la clase A y B es *A(i)+B(i).* De otro modo, el Costo Total es *max(A(i), B(i)),* donde *i<>j.* Cuando un objeto de la clase A y B están almacenados en el mismo sitio, serán enviados al sitio correspondiente en orden secuencial, Por lo tanto el Costo Total de recuperar objetos de la clase A y B es *A(i) + B(i).* [CH-P96]

6.2.1. Obtención del Modelo del Costo de Consulta p/c Ejecución

Como es propuesto en [BE-L96], el proceso de optimizar una consulta basada en el costo (cost-based) es modelada por tres dimensiones principales:

1. El espacio de ejecución,
2. Un modelo del Costo, y
3. La Estrategia de Búsqueda

Como ya se ha mencionado, la dimensión de autonomía que caracteriza a las BDCs en un entorno federado presenta limitaciones en la optimización de la consulta federada, tal es el caso del poco conocimiento de los optimizadores de las BDCs, esto lógicamente repercute en el espacio de ejecución y en la estrategia de búsqueda que a su vez es importante para estimar el costo de la consulta. El problema puede ser descrito abstractamente como sigue [Du-P92]: Dada una consulta Q, un espacio de ejecución E que procesa Q, y una función del costo *c* definido sobre E, encontrar una ejecución *e* en EQ (subconjunto de E que procesa Q) que tenga el mínimo costo: [min $_{e \ni EQ}$ c(e)].

Por lo tanto, el espacio de ejecución se puede definir como el espacio donde se mantiene el modelo de ejecución y definiciones, de una manera abstracta, las ejecuciones alternativas. Un modelo del costo es el cual predice el costo de una ejecución (Algoritmo 7) y la estrategia de búsqueda (Algoritmo 8), la cual es usada para enumerar las ejecuciones y seleccionar la mejor de todas.

Uno de los problemas, como ya se ha venido mencionando es acerca de las BDCs y su poca información que se puede obtener debido a la autonomía y heterogeneidad, para resolver este problema se propone encontrar un **Modelo de Costo Lógico Local** que nos de características de las BDCs. Por lo que se desarrolla un **proceso de calibración** para derivar coeficientes de costo. Lógicamente las formulas del costo derivado pueden ser usadas para estimar el costo de las consultas *ad-hoc* de los usuarios federados.

========== Algoritmo 7 ==========

```
/*Algoritmo para Estimación del Modelo del Costo

Begin (Tree[Nodos],EI,Mapp)                    /* toma como entrada el árbol con nodos
                                                  intermedios y nodos hojas reestructuradas,
                                                  Esquema Integral, Mappings*/
      Procedure Obtiene_Plan(Tree[Nodo]) /*Presenta los planes de ejecución, generados
                                           en fase 2 con heurísticas (ver path expression)
```

```
                                        */
    for(NumPlan=1;NumPlan<=TotalPlan;NumPlan++) do   /*ciclo de procesamiento de
                                        costos de los planes propuestos en Algoritmo
                                        3 */
        Procedure Formula_Costo() /*se aplican las formulas a los operadores
                                        para estimar el costo parametrizado.*/
        Procedure Costo_Metodos() /*Analiza el alcance que tienen los
                                        metodos en los distintos esquemas de
                                        BDCs*/
        Procedure Guarda_Estadísticas()      /*Guarda las estadísticas de costos
                                        para cada Plan de Ejecución en la Base de
                                        Datos de Estadísticas*/
    end_for(NumPlan)
end_Begin
```

========== Fin Algoritmo 7 ==========

6.2.2. Estrategia de Búsqueda

========== Algoritmo 8 ==========

```
/*Algoritmo de Mejor Estrategia
Costo = MaxValor;                       /*Valor de inicio del costo, para
                                        encontrar mejor Plan de Ejecución */
Begin (Tree[Nodos],EI,Mapp)             /* toma como entrada el árbol con
                                        nodos intermedios y nodos hojas
                                        reestructuradas, Esquema Integral,
                                        Mappings */
    for(NumPlan=1;NumPlan<=TotalPlan;NumPlan++) do   /*ciclo de análisis de mejor plan

                                                de ejecución */
        Procedure GetEstadísticas(Costo1)   /*De la base de datos que apoya a la
                                        optimización extraemos los costos
                                        totales del Plan Pn, generado en el
                                        proceso de estimación de la
                                        ejecución.*/
        if Costo1 < Costo Then              /* Encuentra el Mejor Plan */
                Costo = Costo1;
                MejorPlan(Tree[Nodos]) = NumPlan
        End if
    end_for(NumPlan)

end_Begin
```

========== Fin Algoritmo 8 ==========

El ejemplo 1, propuesto al inicio del trabajo, esta condicionado el orden de ejecución de sus subconsultas por la fase de optimización si ésta encuentra Planes de Ejecución que puedan minimizar los costos de recursos presentes y el tiempo de respuesta. Para tal efecto y ejemplificando tomemos en cuenta que la subconsulta 8 y 9 fueron elegidas como primer bloque enviado a las BDCs, bases de datos orientadas a objetos previamente procesar técnicas de calibración mencionadas en modelo del costo, para su posterior unión, y esta unión aplicando las técnicas para mejorar el tiempo de respuesta son enviadas de forma concurrente a las BDCs como lo indica la figura 13.

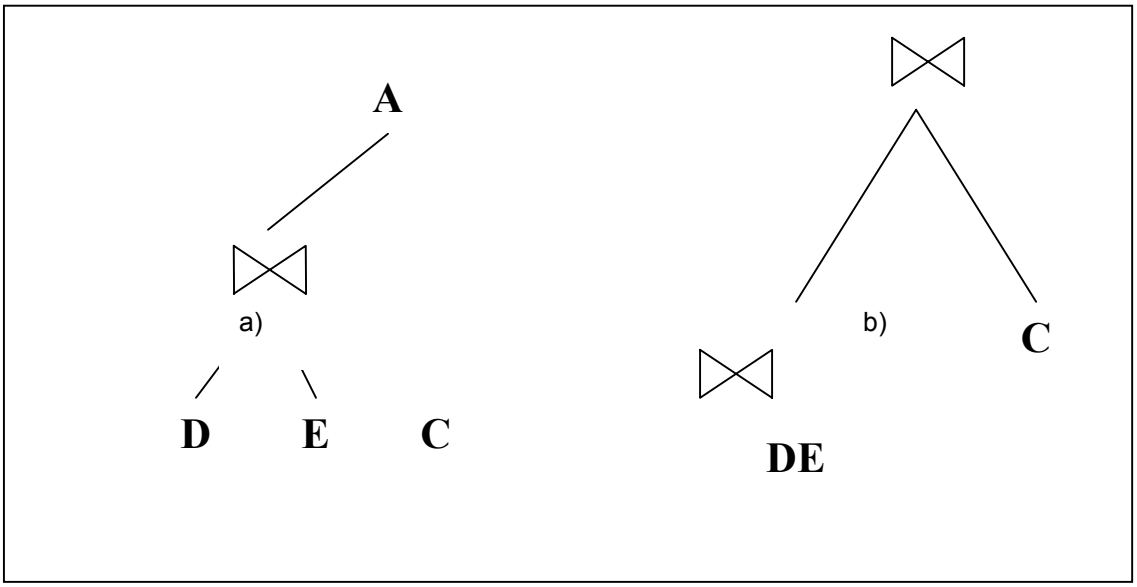


Figura 13. la ilustración muestra el Plan de Ejecución

Cabe mencionar que el ejemplo 1, es muy simple, por lo que la optimización resulto un tanto predecible, pero el procesamiento como se menciono anteriormente, recae en una exhaustiva búsquedas de planes de ejecución en base a predicados predefinidos y analizados (figura 13a), en la técnica de paralelismo y apoyado en el algoritmo 6 obtenemos que la figura 13b) ejecutara concurrentemente 2 visitas a BDCs. Las subconsultas hasta este momento los nombres de los estudiantes de Licenciatura y de Postgrados, con sus respectivos datos, y consecuentemente la unión de los nombres de BDC2 con BDC1, los ***oif*** ya fueron analizados.

6.2.3. Mejorando Tiempo de Respuesta

========== Algoritmo 9 ==========

```
/* Algoritmo Tiempo de Respuesta
/* En este algoritmo se busca aplicar el Paralelismo en las subconsultas , que no son afectadas
por la espera de resultados intermedios, y mejorar el tiempo de respuesta, esto también forma
parte de la Regla 2*/

Procedure Paralelismo (Parámetros del Mejor Plan)
        for (C := 1, TnumEsqC, C++)                    /* Ciclo de reconocimiento de
                                                       subconsultas contra EC */
        for (C1 := 1, TnumClasesC, C1++)               /* Ciclo de reconocimiento de clases
                                                       atómicas de cada clase virtual*/
                seek Nodo3[] (Nodo3[].Padre == C  && Nodo3[].Pos == C1)
                                                       /* Busca Nodo, se analizan nodos
                                                       finales por lo que ven todas las
                                                       subconsultas del Mejor Plan de
                                                       Ejecución */
                if  Paralelismo(Nodo3[].Pos=n) then    /* monitorea cada
                                                       subconsulta, es factible para realizar
                                                       paralelismo y formar parte de Lote
                                                       Concurrente */
                        Save((Nodo3[].Pos=n), Directorio_Est) /* Mantiene en Base de
                                                       datos, para enviar junto con otras y
                                                       mejor tiempo de respuesta */
                End_if
```

```
        end_for (TnumclasesC)
end_for (TnumEsqC)
end_Procedure

Begin (Tree[Nodos],EI,Mapp,MP)          /* Entrada del árbol con nodos
                                           intermedios y nodos hojas
                                           reestructuradas, Esquema Integrado,
                                           Mappings, Mejor Plan MP */
        Get(MejorPlan(Nodo3[])          /*  Obtiene del Mejor Plan de Ejecución
                                           los nodos hojas y aplica Paralelismo,
                                           para ver secuencia de envío al módulo
                                           traductor de consultas */
        Procedure Paralelismo()         /* Aplica Paralelismo */

end_Begin
```

========== Fin Algoritmo 9 ==========

7. Conclusiones y trabajo futuro

Existen muchas formas para descomponer una consulta global y lo mismo sucede para aplicar un determinado plan de ejecución de la consulta. Lógicamente este plan depende en gran medida de la Arquitectura de Ejecución, y por lo tanto de sus operadores para la integración de esquemas y de los algoritmos usados para las correspondencias creados en el Directorio. Algunos SBDF serán por lo tanto más poderosos que otros.

En este trabajo, se aplica una técnica de descomposición de la consulta global que pretende ser eficiente, gracias a la forma en que el Esquema Integrado del modelo BLOOM fue construido, respaldado por sus correspondencias. En ésta los joins explícitos abren la primera fase del proceso, fase que determina la clave principal de la descomposición, ya que las clases virtuales o federadas juegan un papel importante, sino el mas importante. En este documento se presentan técnicas de optimización para una consulta federada en el gestor de consultas del modelo BLOOM. Esta propuesta es para minimizar el costo de los recursos informáticos, así como el tiempo de respuesta. Se plantea resolver problemas que son causados por la autonomía y heterogeneidad de las BDCs. En la fase 2 algunas heurísticas son detalladas para tener los nodos hojas óptimos y poder encontrar los planes de ejecución a evaluar, y en la fase tres vemos que es necesario una estrategia para poder examinar el costo de los planes de ejecución y aplicar por ende un algoritmo que busque el mejor plan, no sin antes procesar estadísticas para un modelo del costo propuesto. Una vez hecho esto nos dimos cuenta que se estaba pasando por alto un punto importante como lo es el tiempo de respuesta, por lo que una estrategia se debe de aplicar, en este caso particular “paralelismo”, para que las subconsultas adecuadas sean procesadas concurrentemente, y poder así salvar un mejor tiempo de respuesta. En este trabajo se pretendió saldar algunos obstáculos pero la creciente autonomía de las BDCs apuntan al desarrollo de mejores técnicas de optimización, es por consiguiente que este trabajo concluye con unas técnicas propuestas y especificas para el modelo BLOOM, tomando como base las ya existentes y algunas extensiones hechas para el BLOOM. Finalmente, la Consolidación de la Respuesta, no menos importante, procesa la integración de respuestas y canaliza el camino de regreso con éxito que dependerá del camino ya formado por el módulo de descomposición.

El futuro para la continuación de este trabajo nota varios retos, como algoritmos suficientes en base a la estrategia presentada, depurándolos y optimizándolos para consultas federadas en el modelo BLOOM y en paralelo investigar sobre nuevas estrategias con eficientes técnicas de optimización, las cuales serán presentadas en próximos trabajos.

8. Referencias:

[AGF-P95] R. Alzahrani, W Gray & N Fiddian. "*A Rule-Based query Processor for a Heterogeneous Object-Oriented Databases Environment – Overview*". Department of Computing Mathematics, University of Wales College of Cardiff, UK. 0-8186-7056-8/95 IEEE.

[ASS-P99] A. Abelló, J. Samos, F. Saltor. "*Benefits of an Object-Oriented Multidimensional Data Model*". Technical Report LSI-99, Dept. Llenguatges I Sistemes Informàtics, Universitat Politècnica de Catalunya.

[AsumanP97] Asuman Dogac, Sena Nural and Fatman Ozcan "*Multidatabase Query Optimization*". Middle East Technical University (METU). Project Number: EEEAG-Yazilim5, by Motorola (USA) and by evgin Holding (Turkey).

[AORS99] A. Abelló, M Oliva, E. Rodríguez y F. Saltor. "*The syntax of BLOOM99 schema"*. Technical Report LSI-99-34-R, Dept. Llenguatges I Sistemes Informàtics, Universitat Politècnica de Catalunya, 1999.

[AORS99-2] A. Abelló, M Oliva, E. Rodríguez y F. Saltor. "*The BLOOM Model Revisited: An Evolution Proposal*". Technical Report LSI-99, Dept. Llenguatges I Sistemes Informàtics, Universitat Politècnica de Catalunya, 1999.

[BE-L96] O. Bukhres & A. Elmagarmid (eds). Object Oriented Multidatabase Systems: Capitulo 5. "*Semantic Heterogeneity in Multidatabase Systems"*. Prentice Hall, 1996.

[BE-L96-Pegasus] O. Bukhres & A. Elmagarmid (eds). Object Oriented Multidatabase Systems: Capitulo 13 "*Query Processing in Pegasus*" Prentice Hall, 1996.

[Fynn_P97] Fynn Kofi. "*A Planner/Optimizer/Executioner for Context Mediated Queries*". Massachusetts Institute of Technology, May 1997. PhD. Thesis.

[CAS-P93] M. Castellanos. "*A methodology for Semantically Enriching Interoperable Databases*". In Worboys & Grundy (eds.) *Advances in Databases*, LNCS 696, Springer, 1994.

[CDN_P92] Carey M, DeWitt, D, Naughton, J., "*The 007 Benchmark*". Computer science departament University of Wisconsin-Madison. Proyecto de DEC, fundado por DARPA contrato número:DAAB07-92-G-Q508, US Army Research Laboratory.

[CORRS-96] B. Campderrich, M. Oliva, E. Rodríguez, F. Saltor, J. Samos, J. Sistac. "*El proyecto BLOOM de investigación en Sistemas de Gestión de Bases de datos Federadas".* B. Campderrich, M. Oliva, E. Rodriguez, L.C. Rodriguez, F. Saltor, J. Samos & J. Sistac. Actas de las Primeras Jornadas en Investigación y Docencia en Bases de Datos, A Coru&ntildea, 1996, pp 272-282. (In Spanish)..

[CU-P96] D. Callen & S. Urban. "*Consolidation of Query Results in Multidatabase Environment: An Object-Oriented Approach*". Department of Computer Science and Engineering, Arizona State University. 0730-3157/96 IEEE 1996.

[Du-P92] Du, W. Krishnamrthy, and Shan, M. "*Query Optimization in Heterogeneous DBMS"*. In Proceedings of International Conference on Very Large Data Bases, Vancouver, Canada, August 1992.

[GeSe-p99] Getta, Janusz, Sedighi, Seyed. "*Otimization Global Query Processing Plans in Heterogeneous and Distributed Multidatabase Systems*". University of Wollongong, Australia. IEEE 0-7695-0281-4/99, 1999.

[GSCP93] M. García-Solaco, F. Saltor y M. Castellanos. "*Discovering Interdatabase Resemblance of Classes for Interoperable DataBases*". In Schek, Sheth, and Czejdo, editors, Proceedings of 3rd Int. Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems, Vienna, pages 26.33. IEEE-CS Press, 1993

[GSCP95] M. García-Solaco, F. Saltor y M. Castellanos. "*Extensional Issues in Schema Integration*". In T. Cheung, editor, Proc. 6th Int. HK DB Workshop: Database Reengineering and Interoperability, pages 243-254, Hong Kong, 1995. City University of Hong Kong.

[GST_P96] Gardarin, G., Sha F., and Tang, Z. "*Calibrating the Query Optimizer Model of IRO-DB, an Object-Oriented Federated Database System*". Proceedings of the 22nd VLDB conference Mumbai (Bombay), India, 1996.

[HK_P99] Hsu Chun, Knoblock, C. "*Semantic Query Optimization for Query Plans of Heterogeneous Multidatabase Systems*". Research reported. Under Grand No. IRI.9313993. National Science fundation. At USC/information science institute and Arizona State University. 22 May 1999.

[IEU-P94] Igras Eugene. "*A framework for query processing in a federated database system: A case study*", http://www.odyssey.maine.edu/gisweb/spatdb/urisa/ur94016.html.

Visita a la página electrónica en Marzo del 2001.

[KC-P95] Jia-Ling Koh & Arbee Chen, "*A Mapping Strategy for Querying Multiple Object Databases With a Global Object Schema*". Department of ComputerScience and Engineering, National Tsing Hua University 0-8186-315/95 IEEE. Taiwan 1995.

[LPL-P96] Ling Liu, Calton Pu, Yooshin Lee, "*An Adaptive Approach to Query Mediation across Heterogeneous Information Sources*", Department of Computing Science, University of Alberta. 0-8186-7505-5/96 IEEE, 1996.

[LSriP93] E. Lim & J. Srivastava, "*Query Optimization and Processing in Federated Database System*", University of Minnesota, MN. Department of Computer Science. ACM 0-89191-626-3/93/0011, USA, 1993.

[LOG_P98] Lu, Ooi, Goh, Cheng_Hian. "*Multidatabase Query Optimization: Issues and Solutions*" National University of Singapore, Dept. Of Information Systems and computer Science. Report 290921.

[ON_P97] Ozcan, F., Nural Sena. "*Dynamic Query Optimization on a Object Management Platform*". Middle East Technical University (METU). Project Number: EEEAG-Yazilim5, by Motorola (USA) and by evgin Holding (Turkey).

[ROSC-P97] E. Rodriguez, M. Oliva, F. Saltor & B. Campderrich; "*On Schema and Functional Architecture for Multilevel Secure and Multiuser Model Federated DB Systems*". In: S. Conrad et al. (eds), Proc. Of the Int'l CAiSE'97 Workshop on Engineering Federated Database Systems (EFDBS'97), pp 93-104. Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, 1997.

[SCI-P99] J. Samos, A. Abello, M. Oliva, E. Rodríguez, F. Saltor, J. Sistac. "*Sistema Cooperativo para la Integración de Fuentes Heterogéneas de Información y almacenes de Datos*". NOVATICA nov/dic. 1999 nº. 142.

[SCGP91] F. Saltor, M.Castellanos, and M. García. "*Suitability of data models as canonical models for federated databases*". ACM SIGMOD Record, 20(4):44-48, 1991.

[SCGP91-2] F. Saltor, M.Castellanos, and M. García. "*The Development of Semantic Concepts in the BLOOM Model using and Object Metamodel*". Technical Report LSI-91-22, Dept. Llenguatges I Sistemes Informàtics, Universitat Politècnica de Catalunya, 1991.

[SCGP93] F. Saltor, M.Castellanos, and M. García. "*Overcoming Schematic Discrepancies in Interoperable Databases*". In D.K. Hsiao, E.J. Neuhold, and R. Sacks-Davis, editors, Interoperable Database System (DS-5), pages 191-205. North-Holland, 1993.

[SCGP93-2] F. Saltor, M.Castellanos, and M. García-Solaco "*Modelling Specialization as BLOOM semilattice*". In H. Jaakkola, editor, Information Modelling and Knowledge Bases VI *(4th European-Japanese Seminar on Information Modelling and Knowledge Bases*), pages 447-467, Kista, 1994. IOS Press, Amsterdam 1995.

[SL-P90] A.P. Shet & J.A. Larson, "*Federated Database System for Managing Distributed, Heterogeneous, and Autonomous Databases*". ACM Computing Surveys, vol. 22, no. 3, Sept. 1990, pp. 183-236.

[WC-L95] Won, Kim. Modern Database System, The Object Model, Interoperability , and Beyond. Capitulo 27, "*Query Processing in Multidatabase systems*". Won Kim (Editor), ACM Press NY, NY, 1995.

[YW-L97] C. Yu & W. Meng. Principles of Database Query Processing for Advanced Aplications. Morgan Kaufmann Publishers, Inc. San Francisco, California. 1997