Instituto Politécnico Nacional

Centro de Investigación en Computación

"Aplicación de técnicas de optimización para la generación de planes de ejecución de consultas hacia bases de datos remotas"

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

ING. RODOLFO NAVARRO ZAYAS

DIRECTOR DE TESIS: **M. en C. ALEJANDRO BOTELLO CASTILLO**

MÉXICO, D.F. **AGOSTO 2014**

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 05 del mes de junio de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Aplicación de técnicas de optimización para la generación de planes de ejecución de consultas hacia bases de datos remotas"

Presentada por el alumno:

NAVARRO	ZAYAS	RODOLFO
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

A	1	2	0	4	1	7

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director de Tesis

M. en C. Alejandro Botello Castillo

Dr. Adolfo Guzmán Arenas

Dr. Gilberto Lorenzo Martínez Luna

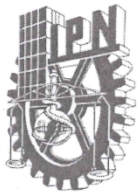
M. en C. Sandra Dinora Orantes Jiménez

Dr. José Giovanni Guzmán Lugo

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION



INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS.

En la Ciudad de **México**, D.F. el día **5** del mes de **Junio** del año **2014**, el que suscribe **Rodolfo Navarro Zayas** alumno del Programa de **Maestría en Ciencias de la Computación**, con número de registro **A120417**, adscrito al **Centro de Investigación en Computación**, manifiesto que es el autor intelectual del presente trabajo de Tesis bajo la dirección del **M. en C. Alejandro Botello Castillo**, y cede los derechos del trabajo titulado ***"Aplicación de técnicas de optimización para la generación de planes de ejecución de consultas hacia bases de datos remotas"***, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a las siguientes direcciones rnavarro_a12@sagitario.cic.ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Rodolfo Navarro Zayas

Resumen.

Un sistema mediador es aquel que permite llevar a cabo la integración de datos que residen en fuentes de información generalmente heterogéneas – que no son del mismo sistema y/o modelo de datos - y remotas, y para ello utiliza una representación unificada de los datos (denominada esquema mediado). Este esquema es utilizado por el usuario para formular consultas particulares al sistema mediador, sin que el usuario conozca la ubicación física de los datos, ni los esquemas de las fuentes en las que estos se encuentran alojados. Estos aspectos deben ser resueltos por el sistema mediador al tiempo de ejecutar la consulta.

En este trabajo se presenta una propuesta de solución para la generación de planes de ejecución de consultas hacia bases de datos remotas, que podrá ser incorporada por el módulo optimizador de un sistema mediador, y en donde el enfoque propuesto inicia recibiendo la consulta del usuario (que inicialmente se encuentra en términos del esquema mediado) y procede a: a) hacer la selección de fuentes relevantes (aquellas que cooperan con respuestas trascendentes), b) la descomposición de la consulta en expresiones que están en términos de los esquemas locales (denominadas consultas objetivo), así como, c) generar la definición del orden de las operaciones de reunión, buscando que este orden proporcione agilidad al procesamiento de la consulta en la integración.

Para determinar el orden de ejecución, se aplican algunas fórmulas de optimización para estimar el número de registros resultantes de las reuniones implicadas en la consulta, tomando en cuenta los valores de selectividad (número de tuplas que cumplen un predicado sobre el total de tuplas), la cardinalidad (los valores que aparecen en un atributo) y los tamaños de las tablas (número total de tuplas) involucradas; posteriormente se utilizan estos datos en un algoritmo *Greedy* cuya función de progreso consiste en dar una mayor prioridad a la ejecución temprana de las operaciones que generan una cantidad menor de tuplas, con el fin de evitar o postergar tanto como sea posible las operaciones que generan más tuplas y por consiguiente reducir el tiempo de respuesta del sistema para resolver la consulta del usuario.

Abstract.

A mediator system performs the integration of data which is located in remote sources, these sources usually are heterogeneous (it means that sources do not belong to the same system or that the sources use different data models to store the information). In order to handle these differences the mediator uses a unified data representation (called mediated schema), this representation is used by the user to express their queries to the system, therefore the user does not need to know the location of the data or the schemas of the sources where the data is stored, These aspects are handled by the mediator system at query execution time.

In this text is presented an approach to generate optimal query execution plans through the access to remote databases, this approach must be able to be incorporated in a query optimizer module of a mediator system. This module receives the user query (which initially is expressed in terms of the mediated schema) and then: a) perform the selection of the relevant sources (those which provide meaningful results), b) do the query decomposition of expressions in terms of local schemas (called target queries) and c) determine the execution order of joins operations, aiming to provide agility to query execution process.

To determine the execution order, some optimization formulas are performed for estimating the number of records resulting from the join operations involved in the query, considering the selectivity values (number of tuples that satisfy a condition) cardinality (the values that appear in an attribute) and the sizes of the tables (total number of tuples) involved; then these data is used for a *Greedy* algorithm whose progress evaluation function gives greater priority to the early execution of operations that generate fewer tuples, in order to avoid or delay as much as possible the operations that generate more tuples and thus reduce the response time of the system to resolve the user query.

Agradecimientos.

Mi director de tesis: M. en C. Alejandro Botello Castillo por sus enseñanzas, su guía, tiempo y continuo apoyo durante el desarrollo de esta tesis, la terminación de este trabajo fue gracias a usted.

Mis sinodales: Dr. Adolfo Guzmán Arenas, Dr. Gilberto Lorenzo Martínez Luna, M. en C. Sandra Dinora Orantes Jiménez y el Dr. Giovanni Guzmán Lugo por los conocimientos que me brindaron y por su constante apoyo durante la realización de este trabajo.

Mi asesor anfitrión: Dr. Satoshi Fujita por brindarme su amabilidad y apoyo durante mi estadía en la universidad de Hiroshima.

Mis profesores: Dr. Rolando Menchaca Méndez y Dr. Juan Luis Díaz de León Santiago por las importantes enseñanzas que me impartieron.

A la profesora: Dra. María Elena Acevedo Mosqueda por ser un ejemplo a seguir en el área de la investigación y por su amistad.

Mis amigos del CIC: Yair, Elías, Luis, Daniel, Álvaro, Niels, Víctor, Araceli, Eric, Memo e Iliac, por lo que he aprendido de ustedes, los momentos que hemos compartido y por la amistad que me han brindado en esta etapa tan importante.

Mis amigos: Aarón, Isaí, Fernando, David, Elizabeth, Arturo y Luis Gustavo por ser mis amigos de toda la vida, por haber estado siempre para mí, por su apoyo y cariño incondicional.

Al Centro de Investigación en Computación: Por la oportunidad otorgada al creer en mí y por permitirme desarrollarme en el ámbito de la investigación.

Al personal administrativo del CIC: Dr. Víctor Hugo Ponce Ponce, M. en C. Santiago Jaime Reyes Herrera, Sra. Silvia Arteaga, Lic. Lourdes y al personal del DTE por su gran apoyo.

Al Instituto Politécnico Nacional: Mi Alma Máter, que me abrió sus puertas permitiéndome formar parte de tan importante institución y por ser la institución que me ha formado como profesionista.

Al Consejo Nacional De Ciencia Y Tecnología y a la Secretaría de Ciencia, Tecnología e Innovación del Distrito Federal (SECITI): Por los apoyos provistos durante mi formación profesional y en la realización de los proyectos de investigación.

De todo corazón muchas gracias.

Dedicatoria.

Al creador de todas las cosas, el ser maravilloso que me ha brindado de todas las bendiciones que hoy forman parte de mi vida, aquel que ha cuidado mis pasos desde antes que pudiera darlos y quien ha demostrado su inmenso amor hacia mi durante cada día de mi vida, por ello primeramente deseo dedicar este trabajo a Dios.

De igual forma, deseo dedicar este trabajo a mi familia; las personas más importantes en mi vida, aquellas que siempre me han brindado su amor, apoyo y cariño. Mi papá por ser mi ejemplo a seguir y haberme inculcado buenos hábitos y valores, a mi mamá que es mi mejor amiga, cuyos consejos y enseñanzas son invaluables para mí, a mi dulce hermana que siempre ha velado por mí y me ha apoyado en cada decisión que he tomado y a mi hermano que he admirado desde que tengo uso de razón y de quien siempre puedo obtener un comentario crítico y objetivo.

Tabla de contenido.

Índice de figuras.

Índice de tablas.

Glosario de términos.

Adaptador (Wrapper): Es un componente vinculado al proceso de integración de datos, el cual puede convertir los datos de una fuente de información de un modelo de datos a otra representación.

Álgebra relacional: es un lenguaje de consulta procedimental. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación. Las operaciones fundamentales del álgebra relacional son selección, proyección, unión, diferencia de conjuntos, producto cartesiano y renombramiento. Además de las operaciones fundamentales hay otras operaciones, por ejemplo, intersección de conjuntos, reunión natural, división y asignación.

Algoritmo Bucket: Es un algoritmo usado por sistemas de integración de información para realizar el desdoblamiento de la consulta de un esquema mediado en expresiones en términos de las vistas a las que tiene acceso el sistema, su funcionamiento está dividido en tres etapas: construcción de *buckets* (contenedores) por cada submeta, la combinación de los elementos contenidos en cada uno y la validación de resultados.

Algoritmo *Greedy*: También conocido como algoritmo voraz, devorador o goloso, es un tipo de algoritmo que con el fin de resolver un determinado problema, sigue una heurística consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima. Este esquema algorítmico es el que menos dificultades plantea a la hora de diseñar y comprobar su funcionamiento, normalmente es utilizado en problemas de optimización.

Almacén de datos: Una colección de datos clasificada por temas, integrada, variable en el tiempo y no volátil que se utiliza como ayuda al proceso de toma de decisiones por parte de quienes dirigen una organización.

Árbol de consulta: Es una estructura de árbol que corresponde a una expresión de algebra relacional en el que las tablas se representan como nodos hojas y las operaciones del algebra relacional como nodos intermedios.

Atributo: También denominado como columna o campo, es un conjunto de valores de datos de un simple tipo particular, uno por cada fila de la tabla. Los atributos proporcionan la estructura según la cual se componen las filas. Muchos consideran más correcto usar el término campo (o valor de campo) para referirse específicamente al simple elemento que existe en la intersección entre una fila y una columna (atributo).

Base de datos: Es una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, que están diseñados para satisfacer las necesidades de información de una organización.

Base de datos distribuida: Una colección lógicamente interrelacionada de datos compartidos (junto con una descripción de estos datos) que se encuentran físicamente distribuidos por una red de comunicaciones.

***Bottom-up*, programación:** Enfoque de la programación en la que todos los problemas que puedan ser necesarios se resuelven de antemano y después se usan para resolver las soluciones a problemas mayores. Este enfoque es ligeramente mejor en consumo de espacio y llamadas a funciones, pero a veces resulta poco intuitivo encontrar todos los subproblemas necesarios para resolver un problema dado.

Cardinalidad de un atributo: Es el número de valores distintos que alberga.

Cardinalidad de una relación: Es el número total de tuplas que contiene.

Consulta: es una forma de buscar, encontrar y exhibir determinada información, extrayéndola del cúmulo de datos que almacena la base.

Dominio: Conjunto de valores permitidos para uno o más atributos.

Esquema mediado: Es la representación unificada y conciliada de un conjunto de esquemas de fuentes de datos autónomas, es definida por el administrador de un sistema de integración y en ella se omiten los detalles referentes al estado físico de los datos.

Equipo mainframe: Es una computadora grande, potente y costosa usada principalmente por una compañía para el procesamiento de una gran cantidad de datos; por ejemplo, para el procesamiento de transacciones bancarias.

ETL, Proceso: Siglas que provienen de las palabras en inglés *Extract-Transform-Load* que significan Extraer, Transformar y Cargar. ETL es el proceso que organiza el flujo de los datos entre diferentes sistemas en una organización y aporta los métodos y herramientas necesarias para mover datos desde múltiples fuentes a un almacén de datos, reformatearlos, limpiarlos y cargarlos en otra base de datos, o almacén de datos.

Factor de selectividad de una condición: Valor que indica la proporción de tuplas en una base de datos que satisface una condición o un conjunto de ellas, se obtiene de la división matemática del número de tuplas que satisfacen la condición sobre el número total de tuplas y siempre es mayor a cero y menor a uno.

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Índice: Una estructura de datos que permite al SABD localizar tuplas dentro de un archivo de una forma más rápida, acelerando así la respuesta a las consultas de los usuarios.

JDBC, conector: Más conocido por sus siglas JDBC (*Java Database Connectivity*), es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos nativa.

Lenguaje de modelado unificado (*Unified Markup Language,* UML.): Es un lenguaje estándar para la especificación, construcción, visualización y documentación de los componentes de un sistema de software. Este lenguaje no prescribe alguna metodología concreta, sino que es flexible y personalizable para adaptarse a cualquier posible técnica y puede utilizarse en conjunción con un amplio rango de procesos de desarrollo y de modelos del ciclo de vida del software.

Llave foránea: Un atributo, o conjunto de atributos, dentro de una relación que se corresponden con la clave candidata de alguna (posiblemente la misma) relación.

Llave primaria: Atributo o conjunto de atributos que identifica de forma univoca cada tupla dentro de una relación.

Metadatos: Son los datos que describen las propiedades o características de otros datos; en caso de las bases de datos, permiten determinar la estructura y el contenido de la base de datos.

Modelo de datos: Una colección integrada de conceptos para describir y manipular datos, las relaciones existentes entre los mismos y las restricciones aplicables a los datos, todo ello dentro de una organización.

Modelo de intercambio de objetos (*Object Exchange Model*, OEM): Es un modelo de datos utilizado para el intercambio de datos semiestructurados entre bases de datos orientadas a objetos. Ha sido usado como modelo de datos elemental en varios proyectos del grupo de bases de datos de Stanford.

Operación diferencia: En álgebra relacional la diferencia entre dos relaciones compatibles A y B, se representa como A MENOS B o de la forma A – B, esta operación produce el conjunto de todas las tuplas t que pertenecen a A y no pertenecen a B.

Operación equi-reunión: La operación de *equi-reunión* es una extensión de la operación reunión natural que permite combinar una selección y un producto cartesiano en una sola operación. Considérense las relaciones *r*(*R*) y *s*(*S*) donde R y S son los esquemas y r y s sus respectivas instancias, y sea θ un predicado de los atributos del esquema $R \cup S$. La operación equi-reunión $r \bowtie_\theta s$ se define de la siguiente manera $r \bowtie_\theta s = \sigma_\theta (r \times s)$.

Operación intersección: En álgebra relacional la intersección de dos relaciones compatibles A y B, se representa como A INTERSECCION B o como A ∩ B, esta operación produce el conjunto de todas las tuplas que pertenecen tanto a A y B. Al igual que en teoría de conjuntos el símbolo ∩ representa la intersección entre dos relaciones.

Operación producto cartesiano: La operación producto cartesiano, denotada por (×), permite combinar información de cualesquiera dos relaciones. El producto cartesiano de las relaciones r_1 y r_2 como $r_1 \times r_2$. Recordando que las relaciones se definen como subconjuntos del producto cartesiano de un conjunto de dominios.

Operación proyección: La operación proyección es una operación unaria que devuelve su relación de argumentos, excluyendo algunos argumentos. Dado que las relaciones son conjuntos, se eliminan todas las filas duplicadas. La proyección se denota por la letra griega mayúscula pi (Π). Se crea una lista de los atributos que se desea que aparezcan en el resultado como subíndice de Π. La relación de argumentos se escribe después de Π entre paréntesis.

Operación reunión natural: La reunión natural es una operación binaria que permite combinar ciertas selecciones y un producto cartesiano en una sola operación. Se denota por el símbolo de reunión (⋈) o algunas veces se utiliza notación como JOIN(A, B). La operación reunión natural forma un producto cartesiano de sus dos argumentos, realiza una selección forzando la igualdad de los atributos que aparecen en ambos esquemas de relación y, finalmente, elimina los atributos duplicados.

Operación selección: El operador de selección elige tuplas que satisfacen cierto predicado, se utiliza la letra griega sigma minúscula (σ) para representar la selección. El predicado aparece como subíndice de σ. La Relación que constituye el argumento se da entre paréntesis después de la σ.

Operación unión: En álgebra relacional la unión de dos relaciones compatibles A y B se representa como A UNION B o A ∪ B, esta operación produce el conjunto de todas las tuplas que pertenecen a A o a B o a ambas. Al igual que en teoría de conjuntos el símbolo ∪ representa aquí la unión de dos relaciones.

Optimización de consultas: La actividad de seleccionar una estrategia de ejecución eficiente para el procesamiento de una consulta.

Procesamiento de consultas: La actividades implicadas en el análisis sintáctico, la validación, la optimización y la ejecución de la consulta.

Procesamiento distribuido: Una base de datos centralizada a la que se puede acceder a través de una red de comunicaciones.

Relación: Esta podría considerarse en forma lógica como conjuntos de datos llamados tuplas y su esquema suele representarse como R (de relación) o (E de entidad) o por alguna letra mayúscula mientras que la instancia de esta suele denotarse por letras minúsculas por ejemplo r o e. Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, esto es, pensando en cada relación como si fuese una tabla que está compuesta por registros (cada fila de la tabla sería un renglón o tupla) y columnas (también llamadas campos).

Relación virtual: El resultado dinámico de una o más operaciones relacionales que operan sobre las relaciones base para generar otra relación. Una relación virtual o vista no tiene por qué existir físicamente de manera necesaria en la base de datos, sino que puede producirse cuando se solicite por parte de un usuario.

Selectividad, índice: Es la relación que hay entre el número de valores distintos en una columna, y el número total de registros. Por ejemplo, si una tabla tiene 1000 registros y una columna indexada de la tabla tiene 950 valores diferentes, entonces la selectividad del índice es 0.95 (950/1000). La peor selectividad es 1, y ocurre cuando hay tantos valores posibles como registros presentes, precisamente en el caso de una clave primaria sobre una columna; la mejor selectividad es la que tiende a 0, y ocurre cuando hay muy pocos valores posibles en relación a los registros presentes, por ejemplo una columna sexo.

Sistema administrador de bases de datos (SABD): Un sistema de software que permite a los usuarios, definir, crear, mantener y controlar el acceso a la base de datos.

Sistema administrador de bases de datos distribuido (SABDD): El sistema software que permite gestionar la base de datos distribuida y hace que dicha distribución sea transparente para los usuarios.

SQL, Lenguaje: El lenguaje de consulta estructurado o SQL (por sus siglas en inglés *Structured Query Language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del

álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma simple información de interés de bases de datos, así como hacer cambios en ellas.

Tupla: También conocida como fila o registro, representa un objeto único de datos implícitamente estructurados en una tabla.

Top down, programación: Enfoque de la programación en la que un problema se divide en subproblemas, y estos se resuelven recordando las soluciones por si fueran necesarias nuevamente. Este enfoque es una combinación de memorización y recursión.

URL: Es una sigla del idioma inglés correspondiente a *Uniform Resource Locator* (Localizador Uniforme de Recursos). Se trata de la secuencia de caracteres que sigue un estándar y que permite denominar recursos dentro del entorno de Internet para que puedan ser localizados; es usada por JDBC para entablar la conexión con una base de datos particular, en donde depende de la sintaxis de esta para establecer la conexión.

XML, tecnología: Más conocido por sus siglas en inglés de *eXtensible Markup Language* (Lenguaje de marcas extensible), es un lenguaje de marcas desarrollado por el W3C (*World Wide Web Consortium*) utilizado para almacenar datos en forma legible. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información.

Capítulo 1 Introducción.

1.1. Antecedentes.

En años recientes se ha replanteado el paradigma operativo empleado para el procesamiento de datos; en lugar de utilizar soluciones basadas en sistemas de administración de base de datos centralizados, que requieren de equipo especializado (como lo son los equipos denominados *mainframe*), se ha optado por el desarrollo de sistemas orientados a la descentralización, mediante la utilización de las redes de comunicaciones, implementando grupos de servidores y la conectividad a través de Internet.

La motivación de estos cambios ha surgido como resultado de múltiples factores: la necesidad natural de distribuir la administración y el control de los datos, para evitar el fenómeno conocido como *"*cuello de botella*"*, que suele ocurrir en un sistema centralizado cuando sus capacidades de procesamiento resultan ser insuficientes para proveer el resultado o rendimiento esperado, así como por el deseo de explotar la existencia de hardware con características más avanzadas y de los nuevos avances tecnológicos en el área de comunicaciones.

A principios de los noventas, se desarrollaron varios prototipos de sistemas de administración de bases de datos distribuidas (SABDD), que eran capaces de realizar el intercambio de información entre bases de datos localizadas en ubicaciones geográficas distintas; no obstante, estos poseían una gran limitante: las fuentes involucradas debían compartir el mismo esquema, lo que resulto ser complicado para poder lidiar con el nivel de heterogeneidad existente en la información manejada por las grandes organizaciones. Como consecuencia de esta situación y con el fin de incorporar las nuevas tendencias manejadas por las organizaciones - que están orientadas a la compartición de información y al empleo de sistemas descentralizados -, se dio lugar a la búsqueda de una alternativa que permitiera llevar a cabo la integración de información proveniente de bases de datos heterogéneas, generando lo que se conoce como los sistemas de integración virtual de datos y los almacenes de datos (*data warehouse*).

Ambos enfoques implican llevar a cabo la combinación de datos provenientes de distintas fuentes pero el proceso que utilizan para llevarla a cabo es distinto; mientras los almacenes de datos inicialmente requieren crear una copia física de los datos, sobre la que después se ejecutarán las consultas del usuario, el integrador virtual de datos lleva a cabo la ejecución en tiempo de consulta, lo que implica la selección de las fuentes relevantes, la extracción de la información desde dichas fuentes y posteriormente su procesamiento.

Recientemente la integración virtual de datos ha captado un creciente interés, debido a que su implementación ofrece mayores beneficios a los brindados por los almacenes de datos. Algunas de estas ventajas son: la integración de fuentes de información que proveen un acceso limitado a sus

datos, permitir al usuario llevar a cabo un análisis de los datos “frescos”, tener la capacidad de manejar varias definiciones del esquema mediado, además de ser una alternativa menos costosa que su contraparte.

La autonomía de las fuentes de datos involucradas conlleva a tener que lidiar con dificultades como las siguientes:

- La heterogeneidad semántica se da cuando hay diferencias en el significado, interpretación y en el uso propuesto de los mismos datos o de datos relacionados; esta situación representa el obstáculo más grande en el diseño de esquemas globales de bases de datos heterogéneas. Por ello los esquemas de las fuentes de datos relacionadas pueden diferir en gran manera en su estructura.

- El conjunto de equipos que almacenan la información que se desea integrar puede implicar un alto grado de heterogeneidad: desde computadoras personales hasta servidores mainframe; así mismo puede variar el software contenido en los equipos.

- Puede requerirse que la integración incluya a dispositivos que no fueron diseñados para una interacción con otras fuentes de datos, por lo que estos pueden carecer de medios que permitan el intercambio de información de las fuentes.

- Pese a utilizar el mismo modelo de datos, los lenguajes y sus versiones varían, por ejemplo en las versiones del SQL que implementan, como el SQL-89, SQL-92, SQL2, y hasta el SQL3; asimismo, cada sistema tiene su propio conjunto de tipo de datos, operadores de comparación, características de manipulación de cadenas de caracteres, etc.

Los sistemas de integración virtual de datos superan estas dificultades utilizando procedimientos especializados para llevar a cabo la integración de los datos, por ejemplo: *Infomaster* [1] provee de un acceso a múltiples bases de datos heterogéneas, que se encuentran localizadas en distintas puntos geográficos y accede a ellas mediante el uso de Internet, dando la ilusión al usuario de estar interactuando con un sistema de información homogéneo de manera local. *InfoSleuth* [2] utiliza la tecnología de agentes, así como ontologías de dominio e intermediación y de cómputo en Internet para realizar la mediación de la interoperación de datos y servicios en un ambiente abierto y dinámico, mientras que *TSIMMIS* [3] se enfoca al empleo de traductores y a la generación de mediadores, lo que reduce de manera significativa el esfuerzo necesario para acceder a nuevas fuentes de origen.

En estos sistemas existe un aspecto que suele ser de vital importancia: la eficiencia del sistema con respecto al proceso de integración, la cual puede valorarse tomando en cuenta distintos criterios (tiempo de respuesta, tiempo de procesamiento, número de accesos a disco, entre otros). En este trabajo se considera el número de tuplas procesadas, siguiendo la premisa de que “un número

menor de tuplas procesadas conllevará a un menor tiempo de respuesta por parte del sistema, para obtener la solución a la consulta del usuario".

Pero el reducir el tiempo de respuesta del usuario no resulta sencillo, dado que los sistemas de integración virtual de datos realizan en tiempo de consulta la creación del plan de ejecución (que especifica la forma en la que extraerán y combinarán los datos de las fuentes locales) y la realización de dicho plan.

1.2. Planteamiento del problema.

Los sistemas mediadores buscan que su funcionamiento pase prácticamente desapercibido para el usuario, brindando la ilusión de contar con un almacén de datos ubicado de manera local, liberando al usuario de tener que conocer la ubicación de los datos y los esquemas a los que pertenecen. Como consecuencia, el mediador es responsable de resolver los aspectos anteriores, por ello en tiempo de consulta se lleva a cabo la creación del plan de ejecución, lo que significa que para que el usuario pueda obtener la respuesta a su necesidad de información, tienen que realizarse las siguientes tareas (VéaseTabla 1.1).

Tabla 1.1 Tareas involucradas en el procesamiento de la consulta del sistema mediador.

Tareas involucradas con la creación del plan de ejecución.	Tareas involucradas con la realización del plan de ejecución.
• La selección de las fuentes relevantes. • La reformulación de la consulta bajo el esquema mediado con respecto a los esquemas locales. • La definición del orden de ejecución.	• La ejecución de las consultas locales. • La transmisión de datos a través de la red de comunicaciones. • La combinación de los resultados de las consultas locales.

Estas tareas se llevan a cabo de manera sucesiva, implicando un tiempo de procesamiento; por consiguiente, el tiempo de espera del usuario para obtener la respuesta a su consulta es igual a la sumatoria de estos tiempos. De esto puede deducirse que resulta imperativo contar con una aproximación que de manera ágil genere planes de ejecución lo suficientemente precisos para resolver las consultas del usuario, lo más eficientemente posible.

Por ello se busca desarrollar una solución a esta problemática, que incorpore los mejores aspectos de los métodos que son usualmente utilizados para la optimización de consultas por los SABD: basados en reglas, basados en costos, heurísticas y estadísticas.

1.3. Objetivos.

1.3.1. Objetivo General.

- Desarrollar una aproximación que permita generar planes de ejecución, que pueda ser incorporada por un sistema de integración virtual de datos para bases de datos relacionales, para definir un orden de ejecución que busque disminuir el tiempo de respuesta del sistema, mediante la incorporación de algunos aspectos de los modelos de optimización de consultas basadas en reglas, costos, heurísticas y estadísticas.

1.3.2. Objetivos Particulares.

- Desarrollar un componente que elabore catálogos, donde se almacenen los metadatos y valores estadísticos (selectividad, cardinalidad y tamaño de tablas) mediante el acceso a las fuentes de datos implicadas.

- Adaptar e implementar procedimientos de los modelos de optimización de consultas basados en reglas, costos, heurísticas y estadísticas para la generación de planes de ejecución en un sistema de integración virtual de datos.

- Diseñar e implementar una propuesta de solución que incorpore los aspectos más relevantes de los enfoques basados en: reglas, costos, reglas, heurísticas y estadísticas, para la generación de planes de ejecución, buscando que su realización beneficie al rendimiento del sistema mediador durante el proceso de integración.

1.4. Justificación.

Los sistemas mediadores permiten llevar a cabo la integración de datos provenientes de fuentes de datos autónomas, de tal manera que este proceso pasa prácticamente desapercibido por el usuario; no obstante, esta practicidad aunada con las tareas de extracción y procesamiento que se realizan después de que el usuario ha expresado su consulta, implica trabajo que tiene que ser realizado por el mediador, lo que afecta directamente al tiempo que el usuario tendrá que esperar para recibir la respuesta a su consulta.

Por consiguiente el presente trabajo busca brindar una solución eficiente para la generación de planes de ejecución en los que se especifiquen las operaciones y el orden para llevarlas a cabo; sin embargo, establecer el orden de ejecución puede resultar complejo, debido a que en ciertas ocasiones el número de posibles alternativas puede ser demasiado elevado y analizar todas ellas podría resultar tardado, por lo que es necesario que el principio de exploración del espacio de posibles soluciones se desarrolle de manera ágil.

1.5. Beneficios esperados.

El desarrollo de este proyecto busca implementar en código la propuesta de solución planteada, de tal manera que esta podrá ser incorporada en el funcionamiento de un sistema.

El resultado que arrojará este desarrollo será un plan de ejecución, como un documento XML, en el que se especificarán las instrucciones a nivel consulta, con respecto a las fuentes de datos locales involucradas en la consulta del usuario y el orden de las operaciones de reunión; dicho orden otorgará una mayor prioridad a la ejecución temprana a operaciones que generen una menor cantidad de registros, con el fin de evitar llevar a cabo procesamiento excesivo, y por consiguiente, reducir el tiempo de espera del usuario para recibir la respuesta a su consulta.

La solución propuesta será codificada en una librería de funciones, esta podrá ser incorporada en otros sistemas y será multiplataforma, contemplando la integración de información proveniente de fuentes de datos contenidas en ambientes operativos distintos.

1.6. Alcances y límites.

Es importante señalar que la solución propuesta se encargará de definir el modo en que se llevará a cabo la integración de los datos, buscando que el procesamiento de la consulta se realice lo más rápido posible; sin embargo, no contempla llevar a cabo la integración de los datos por el mismo.

Antes de que el prototipo pueda recibir las consultas del usuario, es necesario proporcionar previamente en archivos XML (ejemplo de esta estructura se anexa en los apéndices):

- Las correspondencias existentes entre las fuentes de datos que pertenecen al esquema mediado (una correspondencia es la reciprocidad existente entre dos atributos, que pese a pertenecer a esquemas mediados distintos, comparten el mismo dominio).

- El esquema mediado, como la representación unificada de los datos disponibles en las distintas fuentes de información que puede visualizar el usuario.

En tiempo de consulta, el prototipo únicamente aceptará consultas conjuntivas expresadas en términos del esquema mediado y estas tendrán que ser proporcionadas en una representación de tipo XML (ejemplo de esta estructura se anexa en los apéndices).

Posteriormente, mediante el análisis de la consultas y de las características de las fuentes de información participantes, la solución propuesta generará un plan de ejecución global expresado en un archivo XML, y contendrá la información referente a las consultas objetivo necesarias para resolver la consulta del usuario, así como el orden en el que se prevé procesar menos tuplas. Con

esto se buscará reducir el tiempo de procesamiento y por ende el tiempo de respuesta del sistema.

La implementación desarrollada deberá ser capaz de interactuar con distintos Sistema administradores de bases de datos (SABD) ubicados en distintas plataformas. Para cuestiones de pruebas de funcionamiento, se emplearán los sistemas de Oracle, MySQL, DB2 y SQL Server, aunque está abierto a ser probado en otros sistemas similares.

1.7. Organización de la tesis.

En esta tesis se presenta la metodología utilizada, para la implementación de la solución propuesta para la generación de planes de ejecución para un sistema mediador, y se mostrará en el siguiente orden:

En el capítulo 2 se expone el estado del arte sobre los sistemas de integración virtual de datos que existen en la actualidad y los sistemas dedicados a la optimización de consultas, buscando analizar los esfuerzos que se han realizado para el desarrollo de planes de ejecución en sistemas de integración de datos.

En el capítulo 3 se describe el marco teórico, donde se presentan conceptos necesarios en el desarrollo del prototipo.

En el capítulo 4 se expone el análisis y el diseño de la implementación de la solución propuesta, con respecto a las consideraciones descritas en los requerimientos, además se incluyen los diagramas de casos de uso, de secuencia, de actividades y de clases.

En el capítulo 5 se muestran los detalles de la implementación del prototipo, para generar los planes de ejecución en un sistema mediador.

En el capítulo 6 se expone el caso de estudio y los resultados experimentales obtenidos mediante el análisis de los planes de ejecución generados con las consultas de prueba.

En el capítulo 7 se dan las conclusiones del trabajo realizado y las recomendaciones para trabajos futuros.

Capítulo 2 Estado del arte.

2.1. Introducción.

La optimización de consultas siempre ha sido un campo de estudio ampliamente trabajado y como consecuencia de la creciente popularidad de los sistemas de integración virtual de datos, cuya eficiencia depende de la precisión del plan de ejecución, y ha obtenido una mayor atención con el fin de renovarse; por ello en los últimos años se han desarrollado algoritmos basados en una amplia gama de nuevos enfoques, que exploran nuevas alternativas para la creación de planes de ejecución y llevar a cabo la integración de los datos de la manera más eficiente posible.

2.2. Sistemas de integración virtual de datos.

En este apartado se muestran los trabajos que son considerados como antecedentes en la generación de planes de ejecución, por parte de los sistemas integradores de información, y se dará una breve descripción del funcionamiento de algunos de estos sistemas.

2.2.1. The Information Manifold (IM).

El sistema mediador *Information Manifold* [4] permite al usuario visualizar la información contenida en un conjunto de fuentes de datos, que están interconectadas mediante una red de comunicaciones, y le permite expresar sus necesidades de información a través de consultas basadas en un esquema mediado, el cual es generado mediante los metadatos de las distintas fuentes involucradas, que consisten en las descripciones de las fuentes de datos con respecto a sus contenidos y capacidades.

Este sistema utiliza *LAV (Local as View*) para realizar los mapeos y el algoritmo *Bucket* para obtener el resultado de las consultas; además, el proceso de integración utilizado por este sistema contempla la extracción de información ubicada en fuentes de datos distintas y la incorporación de un algoritmo para combinar la información proveniente de las fuentes de datos involucradas; así, los algoritmos utilizados garantizan encontrar las fuentes de datos relevantes para las consultas del usuario y con ello explotar la información contenida en las fuentes de datos locales.

2.2.2. Infomaster.

El sistema *Infomaster* [1] es una herramienta de integración de información desarrollada en la Universidad de Stanford, que proporciona de acceso a los datos ubicados en fuentes de datos heterogéneas, dando la ilusión de contar con un sistema de información centralizado y homogéneo.
Esta herramienta utiliza LAV (*Local as View*), para el almacenamiento de los mapeos de los metadatos y emplea el formato de intercambio de conocimiento (*Knowledge Interchange Format*); además es capaz de utilizar eficientemente las restricciones de integridad y la información de la completitud local para llevar a cabo la etapa de optimización de una consulta, lo que ha quedado demostrado al ser probado en una amplia variedad de áreas de estudio incluyendo las siguientes: ingeniería, logística y el comercio electrónico.

El funcionamiento de este sistema puede ser descrito en dos etapas: planificación y ejecución

La etapa de planificación toma por entrada la consulta del usuario y la información referente a las restricciones de integridad para proporcionar como resultado un plan de ejecución.

La etapa de ejecución recibe como entrada el plan creado y se encarga de llevar a cabo la extracción y combinación de los datos ubicados en las fuentes de datos locales.

A continuación se describe de manera simplificada los pasos que lleva a cabo el algoritmo de planificación:

1. **Simplificación:** En este paso, el sistema reestructura cada campo *atom (*expresión de la forma p (t_1,..., t_n), donde p es un predicado enésimo y cada t_i es una constante o una variable*)* de la consulta mediada usando la definición del predicado asociado, repitiendo este proceso hasta que se obtiene una expresión en términos de alguna de las fuentes locales. La terminación de este algoritmo está asegurada debido a la naturaleza no recursiva de las definiciones utilizadas.

2. **Extracción:** Por cada consulta generada en términos de alguna de las fuentes de datos locales y mediante el conjunto de definiciones, se lleva a cabo la extracción de todas las conjunciones mínimas de los campos compatibles que son contenidos por la consulta.

3. **Minimización conjuntiva:** En esta etapa se elimina cualquier elemento conjuntivo redundante.

4. **Minimización disyuntiva:** En este paso se elimina cualquier elemento disyuntivo redundante.

5. **Agrupamiento:** Finalmente son agrupados los elementos conjuntivos y disyuntivos restantes, con respecto a aquellos que pueden ser respondidos por la misma fuente.

Cabe aclarar que por simplicidad, los pasos anteriormente mencionados se muestran de manera secuencial, lo que en la práctica puede variar un poco debido a que algunos pasos son intercalados; por ejemplo, la minimización conjuntiva es intercalada con el paso de extracción, lo que permite reducir de alguna manera el procesamiento realizado.

2.2.3. Proyecto SIMS.

SIMS [5] es un sistema mediador, cuyo objetivo es integrar información ubicada en distintas fuentes de datos, buscando que esto ocurra sin que el usuario se percate de ello, para ello la información de las distintas fuentes de datos es representada en un esquema mediado.

El principio operativo utilizado por este sistema para resolver la consulta del usuario (que se encuentra expresada en el esquema mediado) consiste inicialmente en identificar las fuentes que son relevantes para resolver la consulta, lo cual se logra mediante el acceso a la información de las fuentes de datos para buscar información relevante.

Este software se especializa en resolver consultas basadas en un único dominio de aplicación y buscar proveer de acceso a las fuentes de datos definidas como elementos del mismo dominio; por ello los modelos del dominio de aplicación son definidos utilizando una base de conocimiento de terminología jerárquica, en la que los nodos representan cada una de las clases de objetos y las aristas entre los nodos constituyen las relaciones entre objetos. Sin embargo, las clases definidas en el dominio del modelo podrían no corresponder exactamente con los objetos descritos en alguna de las fuentes de datos locales, porque el modelo de dominio es una descripción de alto nivel del dominio de la aplicación.

Las consultas recibidas por este sistema están expresadas en término del modelo de dominio y son reformuladas por el sistema en consultas locales, para que puedan ser resueltas por las fuentes de datos a las que se tiene acceso.

El módulo optimizador se auxilia de algoritmos especializados para mejorar rendimiento del sistema mediante la optimización de las consultas locales; no obstante, el aspecto más costoso en el procesamiento de una consulta en un sistema de integración de datos virtual recae en el procesamiento de datos temporales, por este motivo el módulo optimizador se basa en la premisa de la posibilidad de optimizar el orden de ejecución de las consultas locales, de tal manera que solo se recuperarán los datos que cumplieran las condiciones expresadas en etapas posteriores de filtrado, lo que implicaría que los datos temporales se verían reducidos.

La etapa de planificación es similar a la realizada por los SABD, con la diferencia de que para la creación del plan de ejecución se utiliza un planificador basado en herramientas de inteligencia artificial. Mediante estas herramientas, el módulo provee de un gran nivel de flexibilidad, que sobrepasa a las funcionalidades provistas por los SABD comunes.

El planificador permite la construcción del plan de ejecución tomando en cuenta la disponibilidad de las fuentes de datos y al estar integrado al módulo de ejecución, permite que el sistema modifique de manera dinámica a algunas partes de la consulta, en caso de algún posible fallo en la comunicación con la fuente de datos, y con ello continuar la ejecución de las demás consultas locales contenidas en el plan de ejecución.

2.2.4. The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS).

TSIMMIS [3] es un sistema mediador desarrollado por la universidad de Stanford, que busca proveer al usuario de una ágil integración de la información ubicada en fuentes de datos heterogéneas, contenida en un modelo de datos estructurado o semiestructurado.

La meta buscada por este sistema no consiste en realizar la integración de información de manera completamente automática, al esconder los detalles de las diferencias existentes entre las fuentes de datos locales que son independientes, sino que busca proporcionar un *framework* y herramientas que permitan a las personas (usuarios finales o personal del equipo de toma de decisiones) llevar a cabo el procesamiento e integración de información en sus actividades.

Este sistema incorpora los componentes clave necesarios en una típica federación de bases de datos: un componente encargado de extraer las propiedades de fuentes de datos heterogéneas, un mecanismo que traduce la información en un modelo común de objetos, un elemento que combina información de varias fuentes de datos, y un módulo que responde a las consultas a través de múltiples fuentes de datos.

Los aspectos importantes de TSMMIS son el diseño del modelo común y la capacidad de navegar y combinar objetos; no obstante, este omite la inclusión de un módulo orientado a la optimización de consultas.

El proyecto TSIMMIS desarrolló un modelado auto descriptivo de objetos, que fue denominado como modelo de intercambio de objetos (*Object Exchange Model, OEM*), que permitió el anidamiento de objetos.

Otro componente importante en la arquitectura TSIMMIS es la administración de las restricciones de integridad que especifican los requerimientos de consistencia semántica en una fuente de datos, y surgen cuando la información reside en sistemas heterogéneos débilmente acoplados.

Cabe señalar que la administración de las restricciones de integridad en un contexto descentralizado y heterogéneo resulta ser un problema más difícil de lidiar que su contraparte centralizada. Este sistema enfrenta dicha problemática, debido a que en un ambiente débilmente acoplado no es posible el garantizar que cada usuario o aplicación perciben datos consistentes cada vez que se interactúa con el sistema, por ello este sistema incorpora un módulo encargado de fortalecer las restricciones de integridad.

2.3. Sistemas optimizadores de consultas.

A continuación se exponen los sistemas optimizadores de consultas que funcionan como complemento de los sistemas integradores de información. Utilizan reglas de transformación y funciones de costo para realizar una exploración del espacio de las posibles soluciones para obtener la respuesta de la consulta del usuario de la manera más eficiente.

2.3.1. EXODUS.

EXODUS [6] es capaz de proveer una solución para una amplia gama de aplicaciones, debido a que no se encuentra limitado a un solo modelo de datos. Con el fin de ser una solución lo suficientemente general, las consultas y los planes de ejecución son manejados en una representación de árbol en memoria.

El sistema *EXODUS* necesita que al momento de recibir la consulta, este posea un archivo que contenga la descripción del modelo de datos. Esta descripción debe contener: el conjunto de operadores, los métodos y las reglas de transformación (que definen las transformaciones del árbol de consulta), así como de reglas de implementación (que definen la correspondencia entre operadores y métodos). El principio operativo que sigue este sistema consiste en utilizar las reglas de transformación para explorar las expresiones semejantes, y seleccionar los métodos para ejecutar las operaciones, de acuerdo a las funciones de costo asociadas a cada una de estas operaciones.

EXODUS fue el primer optimizador extensible que utilizó la optimización *top-down*, además de incorporar una estrategia de búsqueda adaptativa; es decir, que se modifica a sí misma para auto calibrarse tomando como referencia resultados previos.

El modelo de costos empleado por el optimizador describe el costo total de un árbol de consulta, como la suma del costo de cada uno de los métodos en sus planes de acceso, por lo que es posible llevar a cabo una disminución en los accesos de E/S de resultados intermedios realizando ligeras modificaciones en la función de costo, por ello toda la información disponible es transferida como argumentos a las funciones de costo. Una vez que el mejor plan de acceso ha sido encontrado, el optimizador puede omitir las transformaciones restantes; desafortunadamente no resulta tan fácil

el encontrar el plan óptimo, por ello la estrategia manejada por *EXODUS* consiste en mantener al optimizador buscando este plan mientras no se exceda un número máximo de transformaciones, para ello se compara el costo de las transformaciones realizadas con el costo de la mejor subconsulta equivalente encontrada hasta el momento; si esta representa una mejora, entonces la transformación se aplica, de lo contrario se omite y es retirada de las posibles transformaciones. Los desarrolladores de este optimizador, han expresado la importancia que los sistemas de este tipo debe de separar el modelo de datos de los componentes comunes (mecanismos de búsqueda y de soporte).

Los resultados descritos en [6], demuestran que los planes de ejecución obtenidos son competitivos con aquellos producidos mediante las técnicas de búsqueda exhaustiva, que manejaron un tiempo límite de control relativamente pequeño.

2.3.2. Volcano.

El sistema *Volcano* [7] fue desarrollado por el mismo autor de EXODUS y es considerado como su sucesor ya que buscó brindar un mayor nivel de funcionalidad, además de mejorar el rendimiento del proceso de integración particularmente para sistemas de bases de datos orientadas a objetos y para sistemas de bases de datos científicas.

Su motor de búsqueda permite la utilización de modelos de costo no triviales, es decir, considera las propiedades físicas de las entidades involucradas, por ende la exploración del espacio de solución es más eficiente que el manejado por EXODUS. Combina la programación dinámica con la eliminación de ramas del plan de ejecución mediante una guía heurística, denominada programación dinámica dirigida. La estrategia de búsqueda de este sistema es *top-down* y cuenta con una estrategia de control basada en metas; las subconsultas son optimizadas únicamente si existe una garantía de que están involucradas en planes prometedores a ser considerados por la etapa de optimización.

Volcano logró manejar una mayor extensibilidad mediante la incorporación de un módulo encargado de la generación de código fuente para las especificaciones del modelo de datos, usando la encapsulación de los costos, así como de las propiedades lógicas y físicas en los tipos de datos abstractos. Además es capaz de funcionar como una herramienta independiente o en conjunto con otras aplicaciones, permitiendo el uso de heurísticas y modelos de datos semánticos para guiar la búsqueda y omitir la exploración en todo el espacio de búsqueda.

Este optimizador cuenta con un modelo de costo flexible, que permite la generación de planes de manera dinámica, para consultas que no cuentan con una especificación completamente definida.

En [7] se muestra una comparativa de rendimiento entre *EXODUS* y *Volcano* hecha por el autor de ambos sistemas en el que se puede observar que *Volcano* genera planes de evaluación más eficientes con respecto al tiempo, así como en consumo de memoria, y provee una independencia al modelo de datos así como una extensibilidad más natural.

2.3.3. Framework Cascades.

El *framework* de optimización *Cascades* [8] es un esquema extensible de optimización de consultas que logro resolver algunas deficiencias existentes en los optimizadores *EXODUS* y *Volcano*, por ende es considerado como el sucesor de ambos programas, además es importante señalar que los tres sistemas fueron desarrollados por el mismo autor Goetz Graefe.

Este sistema posee una mejora sustancial con respecto a sus predecesores, en cuanto a funcionalidad, facilidad de uso y robustez, sin renunciar a la extensibilidad utilizando la programación dinámica con memorización, demostrando cubrir los requerimientos y demandas de los sistemas de bases de datos comerciales, por lo que fue utilizado como base para el proyecto *Tandem NonStop SQL* de *Microsoft SQL Server*.

Algunas de las ventajas que ofrece el optimizador *Cascades* son:

- Optimización de tareas como estructuras de datos.
- Reglas manejadas como objetos.
- Reglas para colocar de manera apropiada los elementos *enforcers*, tales como operaciones de ordenación.
- Los predicados son manejados como operadores que pueden ser lógicos así como físicos.
- Código más robusto escrito en C++ y una interfaz clara que permite la plena utilización de los mecanismos de abstracción de ese lenguaje.

El algoritmo de optimización utilizado por *Cascades* se divide en varias tareas percibidas como objetos con métodos definidos, y se reúnen en una estructura de tarea la cual se comporta como una pila “ultimo en entrar primero en salir” (*LIFO Last In First Out*). La planificación de una tarea es muy similar a la invocación de una función, para ello la tarea se extrae de la pila y se invoca al método de la tarea. La realización de una tarea puede resultar en más tareas que se colocan en la pila.

Inicialmente el optimizador Cascades realiza una copia de la consulta original en el espacio de búsqueda inicial (este espacio de búsqueda recibe el nombre de memo). El proceso es desencadenado por una tarea de optimización en un grupo superior del espacio de búsqueda inicial, que a su vez desencadena la optimización de los subgrupos más pequeños y así continúa sucesivamente. La optimización de un grupo significa encontrar el mejor plan para el grupo (a esto

se le conoce como objetivo de optimización) y por consiguiente son aplicadas las reglas a todas las expresiones. En este proceso, las nuevas tareas se colocan en la pila de tarea y nuevos grupos así como nuevas expresiones se añaden en el espacio de búsqueda. Después de la tarea de optimizar del grupo superior se completa, lo que requiere que todos los subgrupos del grupo superior completen su optimización, en donde finalmente el mejor plan del grupo superior es encontrado.

Cascades al igual que el optimizador *Volcano* comienza el proceso de optimización del grupo superior y utiliza una estrategia de búsqueda *top-down*. La programación dinámica y memorización se utilizan para optimizar un grupo. Antes de iniciar la optimización de todas las expresiones de un grupo, se comprueba si el objetivo de optimización ya se ha llevado a cabo, de ser así entonces simplemente devuelve el plan obtenido de la búsqueda anterior.

Una diferencia importante entre la estrategia de búsqueda de *Cascades* y de *Volcano*, reside en que *Cascades* únicamente explora un grupo bajo petición mientras *Volcano* siempre genera todas las expresiones lógicas equivalentes de forma exhaustiva durante la fase previa de optimización antes de que la fase actual de optimización comience. El *framework* de optimización de consultas *Cascades* ha sido uno de los principales exponentes de este tipo de sistemas.

Capítulo 3 Marco teórico.

3.1. Introducción.

A continuación se expone la teoría referente a los sistemas distribuidos, las ventajas y desventajas que conlleva trabajar con estos sistemas, así como las alternativas que existen en la actualidad para llevar a cabo la integración de fuentes de datos heterogéneas, las etapas involucradas en el procesamiento de consultas, los elementos considerados en la etapa de optimización de consultas y otros aspectos que son relevantes para la generación de planes de ejecución.

3.2. Sistemas de bases de datos distribuidos.

Como resultado de los avances relacionados a los sistemas de información y al crecimiento de los datos manejados por las organizaciones, se promovió la descentralización de su información generada hacia diversos sitios, que podrían encontrarse de manera local o dispersos en puntos geográficos distintos; de este contexto surgió la necesidad de integrar y compartir los datos operacionales de las organizaciones, por ello se desarrolló el concepto de bases de datos distribuidas (BDD) (Véase Figura 3.1) definida formalmente en [9] como una colección lógicamente interrelacionada de datos compartidos (junto con una descripción de estos datos), físicamente distribuidos por una red informática de interconexión de datos.

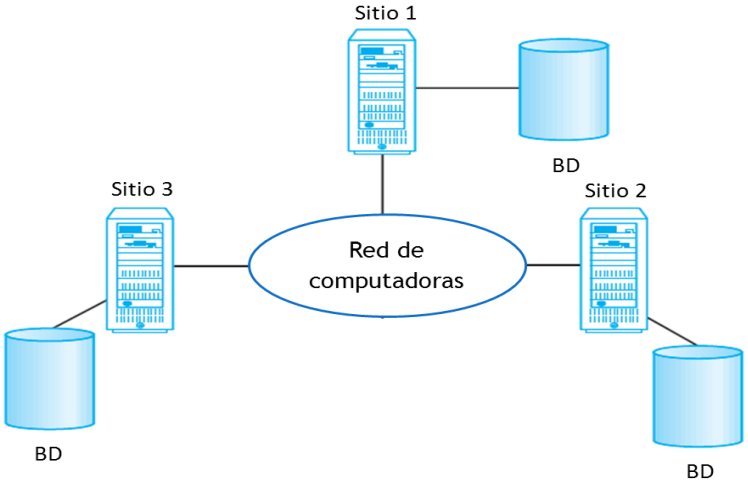


Figura 3.1 Base de datos distribuida.

Para poder administrar una base de datos bajo este esquema es necesario contar con un software especializado denominado Sistema de Administración de Bases de Datos Distribuidas (SABDD) [9], que permite gestionar la base de datos distribuida y hace que las características de distribución, fragmentación, replicación y administración sean transparentes para los usuarios.

El esquema descentralizado es capaz de imitar la estructura organizativa de muchas empresas que están distribuidas de modo lógico en divisiones, departamentos, proyectos, etc., y están distribuidas de modo físico en oficinas, fábricas e instalaciones, manteniendo cada unidad sus propios datos operacionales [10].

Una base de datos distribuida se compone por una única base de datos lógicamente dividida en una serie de fragmentos, donde cada fragmento se almacena en una o más computadoras bajo el control de un SABD independiente y las computadoras involucradas son conectadas mediante una red de comunicaciones. Cada una de las instalaciones es capaz de procesar de forma independiente las solicitudes de los usuarios que requieren acceso a los datos locales (es decir, cada instalación posee cierto nivel de autonomía local) y también es capaz de procesar los datos almacenados en otras computadoras de la red [9].

Tanto la distribución de datos como la distribución de aplicaciones presentan ventajas y desventajas potenciales con respecto a los sistemas centralizados; a continuación se abordarán ambos aspectos:

3.2.1. Ventajas de los sistemas distribuidos.

Estos sistemas ofrecen ventajas como la facilidad de escalabilidad y la capacidad de compartición de datos tanto a nivel local como entre el sistema completo; este esquema permite la autonomía local en los distintos nodos conectados a la red, lo que facilita el control del sistema, dado que pese a la existencia de un administrador global de la base de datos (quien es responsable del sistema completo) es posible que otros administradores se encarguen de gestionar las bases de datos locales.

Un SABDD puede gestionar la fragmentación de la base de datos, manteniendo la información lo más cerca posible del punto donde es más necesaria, por ende se reduce la competición por la CPU y los servicios de E/S, inclusive se atenúan los retardos en el acceso implícito a las redes de área extendida. Cuando se distribuye una base de datos a lo largo de varias localizaciones provoca que se tengan bases de datos más pequeñas, por lo que las consultas realizadas de manera local así como las acciones de acceso a los datos de cada uno de estos sitios poseen un mayor rendimiento debido a su menor tamaño. Por otra parte, el paralelismo puede llevarse a cabo ejecutando múltiples consultas en distintos sitios o descomponiéndola en expresiones más simples, ejecutándose en paralelo, lo que mejora el rendimiento del sistema, y además es posible ocultar al usuario los detalles operacionales de la red, es decir que el usuario ignore la ubicación de cada archivo (tabla, relación, dentro del sistema), siguiendo el principio fundamental de la base de datos distribuida [10]:

“Ante el usuario, un sistema distribuido debe lucir exactamente igual que un sistema que no es distribuido”.

Otra ventaja por la que son caracterizados estos sistemas es el Incremento de la disponibilidad y fiabilidad, lo que es resultado de la capacidad de los SABDD para poder manejar datos replicados y con ello ser capaces de continuar funcionando a pesar de que exista una falla en alguno de sus nodos, asimismo permite el crecimiento modular del sistema dado que el añadir nuevos nodos a la red no afecta las operaciones de otros nodos.

3.2.2. Desventajas de los sistemas distribuidos.

Este tipo de sistemas posee desventajas, por ejemplo: si no se cuenta con una administración adecuada para la replicación de la información, existirá una degradación en la disponibilidad y fiabilidad del sistema; también el hecho de manejar un sistema distribuido implica la necesidad de utilizar un hardware adicional para la implementación de la red de comunicación, así como de costear los gastos propios del sistema de comunicaciones. Con respecto a la seguridad, este tipo de sistemas, además de requerir el control de acceso a los datos replicados en múltiples ubicaciones, requiere que se establezca algún tipo de seguridad a la propia red, así como el control de integridad se vuelve más complicado debido a que los costos de comunicación y de procesamiento requeridos para imponer las restricciones de integridad, los que pueden ser excesivos.

3.2.3. Clasificación de sistemas de BDD.

El termino SBDD hace referencia a todos aquellos sistemas que tienen en común el hecho que tanto como el hardware como el software están distribuidos a lo largo de distintas localizaciones y que están conectadas por algún tipo de red de comunicaciones, por ello a continuación se describirán brevemente los diversos tipos de SBDD así como a los criterios que los distinguen.

El primer factor a considerar es el **grado de homogeneidad del software SABDD**, en donde es el caso de que tanto los servidores (o SABD locales individuales) utilicen el mismo software y también todos los usuarios (clientes) empleen un software idéntico; en cualquier otro caso, es heterogéneo.

Otro factor relacionado con el grado de homogeneidad es el **grado de autonomía local,** en donde el sitio local no es capaz de funcionar de modo aislado, entonces se dice que no tiene autonomía local; en cambio, si se permiten a las transacciones locales un acceso directo a un servidor, entonces podrá decirse que el sistema tiene cierto grado de autonomía local.

En un extremo del espectro de la autonomía tenemos un SABDD, que proporciona la impresión de trabajar con un SABD centralizado, manejando un único esquema conceptual y todos los accesos al sistema se realizan a través de un sitio que forma parte del SABDD, entonces se dice que no hay autonomía local.

En el otro extremo se encuentran los SABDD federados o también conocidos como “sistema de múltiples bases de datos"; en estos sistemas cada servidor es un SABD centralizado independiente y autónomo, que tiene sus propios usuarios locales y transacciones locales, manejando un alto grado de autonomía local.

El termino sistema de bases de datos federados (SBDF) se utiliza cuando existe alguna vista global de la federación de bases de datos que es compartido por las aplicaciones. En cambio, un sistema de "múltiples bases de datos" no cuenta con un esquema global, sino que construye uno de forma interactiva a medida que la aplicación los necesita, cabe aclarar que estos sistemas son híbridos entre los sistemas distribuidos y centralizados y es común referirse a ambos como SBDF en un sentido genérico [11].

El tipo de heterogeneidad presente en los SBDF puede deberse a varios aspectos:

- **Diferencias en modelos de datos:** Las bases de datos de una organización, pueden encontrarse en uno de los distintos modelos de datos que existen: los llamados modelos heredados (red y jerárquico), el modelo relacional, el modelo orientado a objetos e incluso archivos planos. Las capacidades de modelado entre cada aproximación es diferente, por ello resulta desafiante tratar con ellos uniformemente bajo un único esquema global o procesarlos en un único lenguaje; incluso si dos bases de datos utilizan el modelo relacional, la misma información podría representarse de forma diferente en el nombre de los atributos, el nombre de las relaciones, o en el contenido. Por ello es necesario contar con un mecanismo de procesamiento de consultas inteligente que pueda relacionar información basada en metadatos.

- **Diferencias en restricciones:** Las facilidades para especificar e implementar restricciones varían de un sistema a otro. Existen características comparables que deben conciliarse para la construcción de un esquema global. Por ejemplo las relaciones de los modelos Entidad-Relación se representan como restricciones de integridad referencial en el modelo relacional, por ello en este modelo, se deben utilizar disparadores para implementar ciertas restricciones. Para la construcción del esquema global también se debe de lidiar con posibles conflictos entre restricciones.

- **Diferencias en lenguaje de consulta:** Pese a utilizarse el mismo modelo de datos, existen distintas versiones de los lenguajes de consulta, lo que conlleva a diferencias en los tipos de datos, los operadores de comparación, las características de manipulación de caracteres, etc.

- **Diferencia semántica:** Puede darse que existan diferencias en el significado, interpretación y en el uso propuesto de datos semejantes o relacionados; esto es conocido como diferencia semántica y este es resultado de la autonomía de diseño de las fuentes locales.

En los sistemas de bases de datos federado SBDF puede darse que la autonomía de las fuentes se manifieste en distintos grados:

- **La autonomía de comunicación:** Se refiere a la capacidad que tienen las bases de datos para comunicarse con otros sistemas.

- **La autonomía de ejecución:** Esta hace referencia a la habilidad de las fuentes de datos para ejecutar operaciones locales sin interferencias de operaciones externas de otros sistemas y su habilidad para decidir el orden en el que se ejecutarán.

- **La autonomía de asociación:** Esta implica la habilidad de decidir si compartir su funcionalidad (operaciones que soporta) y recursos (datos que gestiona) con otros sistemas semejantes. El mayor desafío en el diseño de SBDF es permitir que las bases de datos puedan operar de manera conjunta, proporcionando los tipos de autonomía anteriormente descritos.

3.3. Sistemas de integración de datos.

Los sistemas de integración son aquellos que se encargan de llevar a cabo la combinación de datos provenientes de distintas fuentes de datos autónomas, proporcionando al usuario una vista unificada de los datos; estos sistemas han captado un creciente interés debido a que su funcionalidad resulta bastante útil, bajo las tendencias actuales.

Existen dos enfoques distintos para llevar a cabo la integración:

- **Física:** Copiando los datos en un repositorio centralizado.
- **Virtual:** Manteniendo la información en las fuentes de datos remotas.

Para llevar a cabo la integración física, es necesario realizar la construcción de un almacén de datos, mientras la otra alternativa consiste en la utilización de un software intermediario (middleware) para realizar una integración virtual o también denominada como aproximación basada en consulta; a continuación se detallarán sobre ambos métodos.

3.3.1. Almacenes de datos (Data warehousing).

En [9] se expone que el concepto original de almacén de datos fue desarrollado por IBM denominándolo “almacén de información” y fue presentado como una solución para acceder a los datos almacenados en sistemas no relacionados. El surgimiento de los almacenes de datos puede atribuirse a dos razones principales: la necesidad de proporcionar una fuente de datos limpia y

consistente para propósitos de apoyo a la toma de decisiones, y la necesidad de hacerlo sin afectar los sistemas operacionales.

Un almacén de datos es un repositorio en el que se combina y almacena la información proveniente de múltiples fuentes bajo un esquema de datos unificado, de tal manera que los datos son almacenados por un largo periodo de tiempo con el fin de utilizarse como historial y posteriormente son analizados mediante complejos análisis estadísticos, utilizando técnicas de descubrimiento de conocimiento con el fin de descubrir patrones entre los datos, que puedan resultar útiles para el apoyo a la toma de decisiones. Los almacenes de datos tienden a ser muy grandes e inclusive pueden tener una taza de crecimiento de hasta el 50% anual, por lo que resulta sumamente complicado perfeccionar su rendimiento.

En [12] se mencionan algunas de las dificultades para emplear esta aproximación:

- Falta de flexibilidad para realizar cambios en la estructura del almacén de datos / escalabilidad del sistema.
- Un tiempo de implementación elevado, debido al tiempo dedicado al diseño del almacén y al proceso de extracción-transformación y carga (*ETL*).
- Mantenimiento del almacén de datos.
- El costo total de la implementación suele ser elevado, debido a que puede necesitar de servidores de uso dedicado.
- El rendimiento del almacén de datos, en donde al manejar una gran cantidad de datos, el tiempo de respuesta del sistema puede ser alto.

Si se realiza una adecuada implementación, las ventajas esperadas según [9] por este tipo de sistemas son:

- **Un alto retorno de inversión:** Pese a ser necesaria una gran cantidad de recursos para implementar un sistema de este tipo, diversos estudios señalan que estos sistemas son altamente redituables; por ejemplo, los resultados del estudio realizado por *International Data Corporation* indican que los retornos de inversión promedio de estos sistemas alcanzaban el 400% en un periodo de tres años.

- **Ventajas competitivas:** Con sistemas de este tipo se permite que las personas responsables de la toma de decisiones tengan acceso a datos que previamente no se encontraban disponibles, y mediante el análisis de estos datos es posible descubrir información importante.

- **Mayor productividad de los responsables de la toma de decisiones:** Un almacén de datos integra la información procedente de múltiples sistemas incompatibles, proporcionando

una representación coherente y simplificada, facilitando la realización del análisis de la información a los responsables de la toma de decisiones.

3.3.2. Sistemas de integración virtual de datos.

Los sistemas de integración virtual de datos suelen operar con un conjunto de fuentes de datos autónomas, por ello se manifiestan en su mayoría las dificultades de los SBDF que se mencionaron anteriormente. Es responsabilidad de este tipo de sistemas solventar las diferencias de los modelos de datos, de los lenguajes de consulta, así como las limitaciones de comunicación, ejecución y asociación.

Algunos ejemplos de sistemas integradores de datos que emplean procedimientos especializados para extraer y combinar la información contenida en fuentes de datos autónomas ubicadas en sitios remotos son: *TSIMMIS* [3], *Information Manifold* [4], *Infomaster* [13], *InfoSleuth* [14] *SIMS* [15] , *GARLIC* [16], por mencionar algunos.

3.3.2.1. Sistemas mediadores.

Los sistemas mediadores han surgido como resultado de la necesidad de explotar la gran cantidad de información que actualmente se encuentra dispersa en fuentes de datos independientes, dado que puede resultar sumamente útil contar con una perspectiva más completa de los datos en aplicaciones corporativas.

Es importante señalar que los mediadores tienen diferencias importantes con respecto las bodegas de datos (Véase Tabla 3.1), pese a que ambos enfoques manejan los datos provenientes de fuentes remotas, persiguen propósitos distintos: los mediadores únicamente ejecutan consultas con fin informativo sobre los datos actuales, mientas que los almacenes de datos lo hacen con tal de realizar todo un análisis.

Una vez que el sistema recibe consulta del usuario, el mediador procede a descomponerla en expresiones en términos de los esquemas locales, para que dichas expresiones pueden ser contestadas completamente por las fuentes de datos locales; estas expresiones son denominadas consultas objetivo, dado que se busca que estas puedan ser enviadas a una determinada fuente o en caso de manejar múltiples modelos de datos a un adaptador (*wrapper)*. Los *adaptadores* son utilizados cuando se necesita manejar una representación neutra de los datos contenidos en las fuentes; en muchos sistemas mediadores se toma al modelo relacional como modelo de datos en común, por ende todos los adaptadores deben proporcionar una interfaz SQL, incluso si la fuente de datos subyacente fuera un sistema de archivos.

Tabla 3.1. Diferencias de los sistemas mediadores y los almacenes de datos.

Característica	Sistema mediador	Almacén de datos
Orientación	Transacción	Análisis
Función	Operaciones diarias	Soporte a la toma de decisiones a mediano o largo plazo
Temporalidad de los datos	Actuales	Históricos
Datos utilizados en tiempo de consulta	Datos de las fuentes operativas	Datos respaldados
Tipo de acceso a los datos en tiempo de consulta	Remoto	Local
Privilegio de acceso a los datos	Lectura	Lectura/Escritura
Tipo de consulta permitida	Consulta simple	Consulta simple, Consulta compleja(operaciones agregadas)
Número de registros analizados	Cientos	Millones
Tamaño de las bases de datos manejadas	Orden de Gigabytes	Orden de Terabytes

Para la creación de las consultas objetivo el sistema mediador se auxilia de un catálogo que contiene los mapeos del esquema mediado con respecto a las fuentes locales, los metadatos, las restricciones de integridad, los detalles específicos para tener acceso a las fuentes de datos y los valores estadísticos que describen la distribución de los datos (cardinalidad y selectividad).

Posteriormente se crea un plan de ejecución, en el que se especifican las operaciones que se realizarán en las fuentes de datos locales y el orden en el que se ejecutarán. En caso de manejar varios modelos de datos, dicho plan tendrá que especificar el adaptador al que se enviará cada una de las consultas objetivo, para que pueden ser contestadas.

Debido a la autonomía de las fuentes de datos involucradas en el proceso de integración, las fuentes locales no tienen la capacidad de comunicarse entre sí [17], ni son capaces de compartir datos o recursos en la ejecución de consultas. El único elemento capaz de combinar los datos provenientes de la colección heterogénea de fuentes es el módulo integrador del sistema mediador [18]; por esta razón el proceso de integración toma como núcleo al sistema mediador, de tal manera que establece una conexión con cada fuente local y una conexión con el usuario, además de encargarse de llevar a cabo todo el procesamiento sobre los datos una vez que se han extraído de las fuentes locales.

3.4. Descripción de los datos.

Para que el sistema mediador lleve a cabo el correcto procesamiento de la consulta, este debe conocer las fuentes a las que tiene acceso, los datos que existen en cada una de ellas y la forma en la que estos se relacionan con los campos del esquema mediado. Las descripciones de las fuentes en un sistema de integración de datos codifican esta información, en donde el componente principal de una descripción de la fuente es llamado esquema de asignación o mapeo, esto es, una correspondencia de los campos que existen en el esquema mediado y los campos de las fuentes locales a los que hacen referencia.

Para ello debe de lidiar con los dos tipos de inconsistencias que pueden aparecer durante el proceso de integración [18].

- **Inconsistencias de esquema:** son las diferencias semánticas entre el esquema global y los esquemas de las fuentes locales.

- **Inconsistencia de los datos:** esta puede deberse a la heterogeneidad de los tipos de datos o de los valores: es posible que en bases de datos distintas se defina el mismo campo, pero usando un tipo de datos diferente, también podría ocurrir que los mismos valores lógicos sean almacenados en formas distintas, por ejemplo el término profesor puede ser almacenado como “Profesor”, “Profe”, “Profr” o “Prof”.

3.4.1. Representación del esquema mediado.

Un sistema de integración de información usando mediador puede representarse mediante una terna <G, S, M> en la que:

- **G:** Representa el modelo del mediador, expresado en un lenguaje L_G sobre un alfabeto A_G. El alfabeto incluirá un símbolo para cada elemento de G (una relación si G es una base de datos relacional, una clase si G es orientada a objetos, etc.).

- **S:** es el modelo origen (el modelo de una fuente de datos a integrar), expresado en un lenguaje L_S sobre un alfabeto A_S. Dicho alfabeto A_S incluirá un símbolo para cada elemento de la fuente.

- **M:** representa las correspondencias (el *matching*) entre G y S, constituido por un conjunto de mapeos.

Por las posibles inconsistencias de esquema, el sistema de integración (el mediador) debe de conciliar estas diferencias mediante una serie de mapeos. En la actualidad existen tres paradigmas distintos que pueden ser ocupados para el mapeo de esquemas [16].

- *Local-as-View (LAV):* El mapeo asocia cada relación de los esquemas locales a una vista del esquema global (Véase Figura 3.2).

Figura 3.2. Local as View.

- *Global-as-View (GAV)*: El mapeo asocia cada relación del esquema global con una vista de los esquemas locales (Véase Figura 3.3).

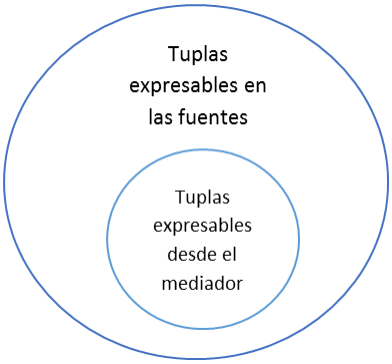


Figura 3.3. Global as View.

- *Global-and-Local-as-View (GLAV):* El mapeo asocia vistas del esquema global a las vistas del esquema global.

3.5. Procesamiento de consultas.

Podría suponerse que el procesamiento de consultas en un sistema de integración de información difiere en su totalidad del desarrollado por un SABD normal; sin embargo todos los lenguajes de consulta (*SQL*, *OQL*, *datalog*, *XQyuery*, etc.) están basados en la algebra relacional (o una versión

extendida de esta) y comparten el mismo objetivo: el encontrar un plan de ejecución eficiente para obtener la respuesta a la necesidad de información del usuario.

3.5.1. Etapas implicadas en el procesamiento de consultas en un SABD.

A continuación se muestran las etapas implicadas en el procesamiento de consultas, como lo ilustra [19] , basándose en el proyecto *Starburst* de IBM (Véase Figura 3.4):

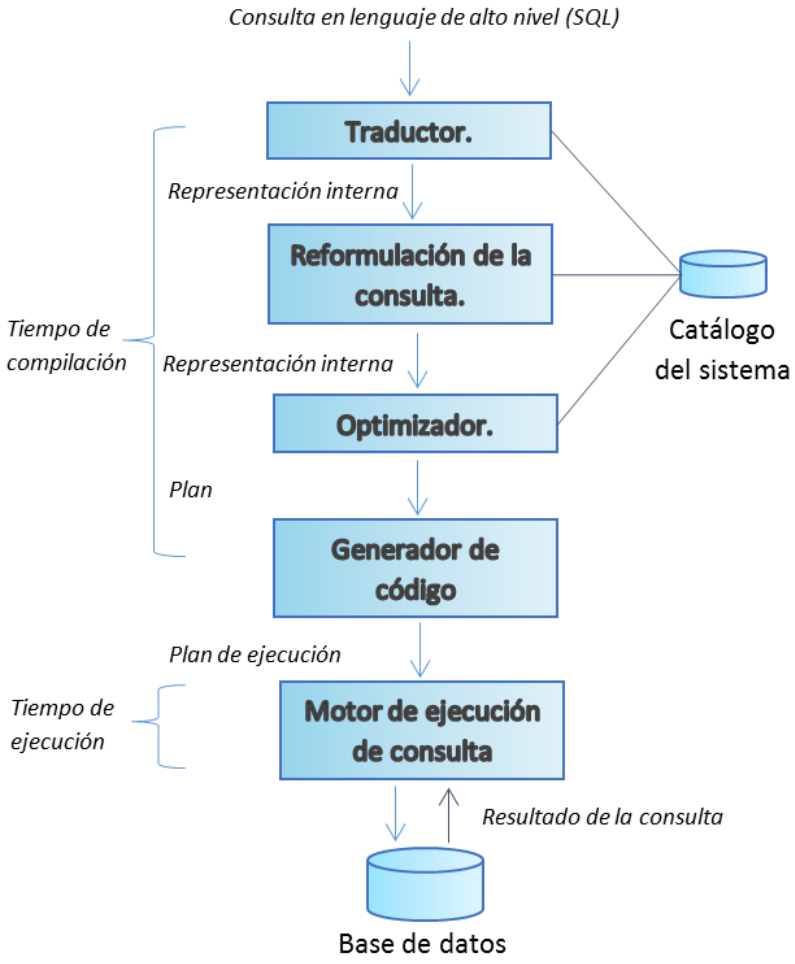


Figura 3.4 Etapas asociadas al procesamiento de consultas en un SABD.

Traductor: Este módulo realiza la transformación de la consulta del usuario, de una expresión en texto plano a una representación interna, que puede ser de tipo árbol, grafo o alguna otra representación conceptual que permita su libre manipulación para las etapas posteriores.

Reformulación de la consulta: En esta etapa, utilizando la información contenida en el catálogo y sin tomar en cuenta el aspecto físico del sistema (tamaño de las tablas, existencia de índices, velocidades de las máquinas, entre otros aspectos) se aplican varias transformaciones a la

expresión de la consulta, como lo son: la eliminación de predicados redundantes y la simplificación de expresiones.

Optimización de consultas: En esta fase se utiliza la información estadística, los metadatos disponibles en el catálogo, considerando el aspecto físico del sistema, para después emplear algoritmos especializados que exploran el espacio de posibles soluciones y con ello generar un plan de ejecución, en el que se especifique: que índices emplear, que métodos utilizar para llevar a cabo las operaciones de la consulta (por ejemplo: *hashing*, *sorting*, *merging*, entre otros), además de establecer el orden en el que se ejecutaran dichas operaciones. En un sistema distribuido también se tiene que determinar los sitios en los que realizarán las distintas operaciones del plan de ejecución, y finalmente el plan es enviado al motor de consulta, que se encarga de llevar a cabo los accesos a las tablas y posteriormente la combinación de resultados intermedios.

Plan: Este elemento especifica de manera puntual el modo en que va ser respondida una consulta, y suele ser expresado en forma de árbol, de tal forma que los nodos son operadores que conllevan una acción o representan una propiedad, y las aristas representan una relación productor-consumidor de los operadores.

Generador de código: Este módulo se encarga de transformar el plan generado en un plan que pueda ser ejecutado, por ejemplo: en *System R,* esta transformación implica la generación de código parecido al ensamblador, buscando llevar a cabo una eficiente evaluación de expresiones y de predicados.

Motor de ejecución de consultas: Este componente se encarga de ejecutar cada operación especificada en el plan de ejecución.

3.6. Optimización de consultas.

La optimización de consultas es el proceso de selección del plan de ejecución más eficiente de entre las muchas alternativas disponibles para el procesamiento de una consulta determinada (Véase Figura 3.5).

Un aspecto de la optimización de las consultas tiene lugar en el nivel del álgebra relacional, donde el sistema intenta hallar una expresión que sea equivalente a la expresión proporcionada, que brinde un rendimiento de ejecución más eficiente. Otro aspecto es la elección de una estrategia detallada para el procesamiento de la consulta, como puede ser la selección del algoritmo que se utilizará para ejecutar una operación, la selección de los índices concretos que se van a emplear, entre otros aspectos.

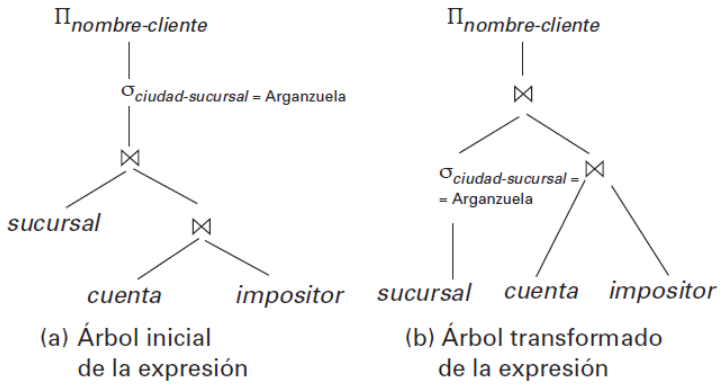


Figura 3.5 Ejemplo de árboles de evaluación de consulta equivalentes.

La diferencia en costo (en términos de tiempo de evaluación) entre una estrategia buena y una mala suele ser importante, y puede ser de varios órdenes de magnitud. Por ello resulta necesario que el módulo optimizador utilice una buena estrategia para encontrar el plan de ejecución de una consulta determinada [20].

3.6.1. Enfoques empleados en la optimización de consultas.

Desde el surgimiento de las bases de datos se han desarrollado distintos enfoques para llevar a cabo la optimización de consultas; a continuación se exponen los que suelen ser considerados por la literatura como las más importantes.

3.6.1.1. Método de optimización de consultas basado en reglas.

Es considerado como un enfoque clásico en la optimización de consultas, la estrategia de optimización basada en reglas sólo tiene en cuenta los caminos de acceso existentes en el esquema interno de la BD (índices primarios, secundarios, *clusters*, rutinas de hashing, etc.), un conjunto predefinido de reglas (métodos alternativos) y la manera de especificar las condiciones en la consulta (constantes, variables, uso de funciones, etc.).

Oracle empleaba este tipo de optimización por defecto hasta la versión 9i, en donde su módulo optimizador se basa en 15 reglas ordenadas según su eficiencia, como se muestra en la Tabla 3.2. El optimizador puede seleccionar la utilización de una ruta de acceso concreta a una tabla solo si la instrucción contiene un predicado u otro tipo de estructura que haga que esa ruta de acceso esté disponible. El optimizador basado en reglas asigna una puntuación a cada estrategia de ejecución utilizando estas clasificaciones y luego selecciona la estrategia de ejecución con la mejor puntuación (en este caso la más baja).

Cuando dos estrategias tienen la misma puntuación, Oracle resuelve el empate formando una decisión que depende el orden en que aparezcan las tablas dentro de la instrucción SQL.

Tabla 3.2 Reglas manejadas por Oracle 9i.

Rango	Reglas
1	Una sola fila basándose en ROWID (el identificador de la fila).
2	Una sola fila basándose en combinación con cluster.
3	Una sola fila basándose en clave *Hash* con clave unívoca o principal.
4	Una sola fila basándose en clave unívoca o principal.
5	Combinación clúster.
6	Clave hash con cluster.
7	Clave indexada con cluster.
8	Clave compuesta (Índice multiatributo).
9	Índice mono-columna.
10	Búsqueda de rango limitado sobre columnas indexadas.
11	Búsqueda de rango no limitado sobre columnas indexadas.
12	Combinación mediante ordenación-mezcla.
13	MAX o MIN de columna indexada.
14	ORDER BY sobre columnas indexadas.
15	Exploración completa de tabla.

Por ejemplo: considere la siguiente consulta sobre la tabla BienesRaicesEnVenta y suponga que existe un índice sobre la llave primaria IdPropiedad, la columna Habitaciones y sobre la columna Ciudad.

- SELECT IdPropiedad.
- FROM BienesRaicesEnRenta.
- WHERE Habitaciones >7 AND Ciudad = “Cuautla”.

En este caso, el optimizador basado en reglas consideraría las siguientes rutas de acceso:

- Una ruta de acceso mono-columna utilizando el índice sobre la columna Ciudad de la condición WHERE (Ciudad = “Cuautla”), esta ruta de acceso tiene rango 9.

- Una exploración de rango no limitado utilizando el índice sobre la columna Habitaciones de la condición WHERE (Habitaciones > 7). Esta ruta de acceso tiene rango 11.

- Una exploración completa de tabla, que está disponible para todas las instrucciones SQL. Esta ruta de acceso tiene rango 15.

Pese a que existe un índice sobre la columna IdPropiedad, el optimizador basado en reglas no lo toma en consideración porque no existe referencia a este campo en alguna de las clausulas WHERE.

Basándose en las rutas previamente mencionadas, el optimizador basado en reglas optaría por utilizar el índice de la columna Ciudad.

3.6.1.1.1. Índices.

El índice de una base de datos es una estructura de datos que mejora la velocidad de las operaciones por medio de identificador único de cada fila de una tabla, permitiendo un rápido acceso a los registros de una tabla en una base de datos. Estos desempeñan el mismo papel que los índices de los libros o los catálogos de fichas de las bibliotecas, por ello suelen denominarse como “caminos de acceso”.

Los índices son usados para encontrar rápidamente los registros que tengan un determinado valor en alguna de sus columnas. Sin un índice, el Sistema administrador de bases de datos (SABD) tiene que iniciar con el primer registro y leer a través de toda la tabla para encontrar los registros relevantes. Aún en tablas pequeñas, de unos 1000 registros, es por lo menos 100 veces más rápido leer los datos usando un índice, que haciendo una lectura secuencial.

Cuando el SABD va a procesar una consulta, examina las estadísticas de los datos involucrados y define la manera en la que se buscarán los datos que deseamos de la manera más rápida. Sin embargo, si una tabla no cuenta con índices, entonces se tendrán que leer todos los registros de la tabla de manera secuencial. Esto es comúnmente llamado un "escaneo completo de una tabla", y es muchas veces algo que se debe evitar.

Los índices son construidos sobre árboles B, B+, B* o sobre una mezcla de ellos, funciones de cálculo u otros métodos.

Deben de evitarse las escaneos completos de tablas por las siguientes razones:

- **Sobrecarga de CPU:** El proceso de revisar cada uno de los registros en una tabla es insignificante cuando se tienen pocos datos, pero puede convertirse en un problema a medida que va aumentando la cantidad de registros en nuestra tabla. Existe una relación proporcional entre el número de registros que tiene una tabla y la cantidad de tiempo que le toma al SABD revisarla completamente.

- **Concurrencia.** Mientras el SABD está leyendo los datos de una tabla, éste la bloquea, de tal manera que nadie más puede escribir en ella, aunque si pueden leerla. Cuando el SABD está actualizando o eliminando filas de una tabla, éste la bloquea, y por lo tanto nadie puede al menos leerla.

- **Sobrecarga de disco.** En una tabla muy grande, un escaneo completo consume una gran cantidad de entrada/salida en el disco. Esto puede hacer más lento de modo significativo el rendimiento del servidor de bases de datos.

Hay dos tipos básicos de índices:

- **Índices ordenados:** Estos índices están basados en una disposición ordenada de los valores.

- **Índices asociativos**: (índices hash). Estos índices están basados en una distribución uniforme de los valores a través de una serie de cajones (*buckets*). El valor asignado a cada cajón está determinado por una función, llamada función de asociación (hash *function*).

Existen varias técnicas de indexación y asociación, cada una con diferentes características, por ello no puede denominarse a alguna como la mejor. Sin embargo, cada técnica es la más apropiada para una aplicación específica de bases de datos.

En [20] señala que cada técnica debe ser valorada según los siguientes criterios:

- **Tipos de acceso:** Los tipos de acceso que se soportan eficazmente. Estos tipos podrían incluir la búsqueda de registros con un valor concreto en un atributo, o buscar los registros cuyos atributos contengan valores en un rango especificado.

- **Tiempo de acceso:** El tiempo que se tarda en buscar un determinado elemento de datos, o conjunto de elementos, usando la técnica en cuestión.

- **Tiempo de inserción:** El tiempo empleado en insertar un nuevo elemento de datos. Este valor incluye el tiempo utilizado en buscar el lugar apropiado donde insertar el nuevo elemento de datos, así como el tiempo empleado en actualizar la estructura del índice.

- **Tiempo de borrado:** El tiempo empleado en borrar un elemento de datos. Este valor incluye el tiempo utilizado en buscar el elemento a borrar, así como el tiempo empleado en actualizar la estructura del índice.

- **Espacio adicional requerido:** El espacio adicional ocupado por la estructura del índice. Como normalmente la cantidad necesaria de espacio adicional suele ser moderada, es razonable sacrificar el espacio para alcanzar un rendimiento mejor.

3.6.1.2. Método de optimización de consultas basado en heurísticas.

El método heurístico de optimización de consultas utiliza reglas de transformación (también conocidas como reglas de equivalencia [20]) para convertir una expresión de algebra relacional en otra forma equivalente que es más eficiente.

Una regla de equivalencia dice que las expresiones de dos formas son equivalentes. Se puede sustituir una expresión de la primera forma por una expresión de la segunda forma, ya que las dos expresiones generan el mismo resultado en cualquier base de datos válida. El optimizador utiliza las reglas de equivalencia para transformar las expresiones en otras equivalentes lógicamente.

3.6.1.2.1. Reglas de transformación para las operaciones de algebra relacional.

La notación que se utilizará para describir las reglas de transformación es la siguiente:

- **E:** Esquema de una relación.
- **σ:** Operación selección.
- **∏:** Operación proyección.
- **⋈θ:** Operación de Equi-reunión.
- **×:** Operación producto cartesiano.
- **∩:** Operación intersección.
- **∪:** Operación unión.
- **∧:** Conjunción de predicados.
- **∨:** Disyunción de predicados.
- **θi:** Son los predicados condicionales.
- **L:** Son los atributos de una relación determinada.

1. Las operaciones de selección conjuntivas pueden dividirse en una secuencia de selecciones individuales. Esta transformación se le denomina como cascada de σ.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. Las operaciones de selección son conmutativas.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. Sólo son necesarias las últimas operaciones de una secuencia de operaciones de proyección, las demás pueden omitirse. Esta transformación también puede denominarse cascada de ∏.

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_1}(E))\ldots)) = \Pi_{L_1}(E)$$

4. Las selecciones pueden combinarse con los productos cartesianos y con equi-reunión.
   a. $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$ *(Equi-reunión)*

b. $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

5. Las operaciones de Equi-reunión son conmutativas.

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

Realmente, el orden de los atributos es diferente en el término de la derecha y en el de la izquierda, por lo que la equivalencia no se cumple si se tiene en cuenta el orden de los atributos. Se puede añadir una operación de proyección a uno de los lados de la equivalencia para reordenar los atributos de la manera adecuada, pero por simplicidad se omite la proyección y se ignora el orden de los atributos en la mayor parte de los ejemplos.

Dado que el operador de reunión natural es simplemente un caso especial del operador de equi-reunión; por lo tanto, las reuniones naturales también son conmutativas.

6. Asociatividad de las reuniones.

   a. Las operaciones de reunión natural son asociativas:

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

   b. Las equi-reuniones son asociativas en el siguiente sentido:

$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$$

Donde θ_2 implica solamente atributos de E_2 y de E_3. Cualquiera de estas condiciones puede estar vacía; por tanto, se deduce que la operación producto cartesiano (x) también es asociativa. La conmutatividad y la asociatividad de las operaciones de reunión son importantes para la reordenación de las reuniones en la optimización de las consultas.

7. La operación de selección se distribuye por la operación de Equi-reunión bajo las dos condiciones siguientes:

   a. Se distribuye cuando todos los atributos de la condición de selección θ_0 implican únicamente los atributos de una de las expresiones (por ejemplo, E_1) que se están reuniendo.

$$\sigma_{\theta_0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_{\theta} E_2$$

   b. Se distribuye cuando la condición de selección θ_1 implica únicamente los atributos de E_1 y θ_2 implica únicamente los atributos de E_2.

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta} (\sigma_{\theta_2}(E_2))$$

8. La operación proyección se distribuye por la operación de Equi-reunión bajo las condiciones.

a. Sean L_1 y L_2 atributos de E_1 y de E_2, respectivamente. Supóngase que la condición de reunión θ_1 implica únicamente los atributos de $L_1 \cup L_2$. Entonces:

$$\prod_{L_1 \cup L_2} (E_1 \bowtie_\theta E_2) = (\Pi_{L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))$$

b. Considérese una reunión $E_1 \bowtie_\theta E_2$. Sean L_1 y L_2 conjuntos de atributos de E_1 y de E_2, respectivamente. Sean L_3 los atributos de E_1 que están implicados en la condición de reunión θ, pero que no están en $L_1 \cup L_2$, y sean L_4 los atributos de E_2 que están implicados en la condición de reunión θ, pero que no están en $L_1 \cup L_2$. Entonces:

$$\Pi_{L_1 \cup L_2} (E_1 \bowtie_\theta E_2) = \Pi_{L_1 \cup L_2} ((\Pi_{L_1 \cup L_3} (E_1)) \bowtie_\theta (\Pi_{L_2 \cup L_4} (E_2)))$$

9. Las operaciones de conjuntos unión e intersección son conmutativas, pero la diferencia de conjuntos no es conmutativa.

$$E_1 \cup E_2 = E_2 \cup E_1$$

$$E_1 \cap E_2 = E_2 \cap E_1$$

10. La unión y la intersección de conjuntos son asociativas.

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

11. La operación de selección se distribuye por las operaciones de unión, intersección y diferencia de conjuntos.

$$\sigma_P (E_1 - E_2) = \sigma_P (E_1) - \sigma_P (E_2)$$

12. La operación de proyección se distribuye por la operación unión.

$$\prod_L (E_1 \cup E_2) = (\Pi_L (E_1)) \cup (\Pi_L (E_2))$$

3.6.1.2.2. Estrategias de procesamiento heurísticas.

Los SABD suelen auxiliarse de algunas estrategias heurísticas para determinar las estrategias de procesamiento de las consultas; a continuación se examinarán algunas reglas heurísticas convenientes que pueden aplicarse para procesar consultas [9].

1. *Realizar las operaciones de selección lo antes posible.*

La selección reduce la cardinalidad de la relación y como consecuencia también se disminuye la cantidad de procesamiento en futuras operaciones que empleen esos resultados, por esta razón es conveniente utilizar la transformación cascada en las operaciones de selección y las reglas referentes a la conmutatividad de la selección con las operaciones unarias y binarias,

con el fin de mover las operaciones de selección lo más abajo posible en el árbol de consulta(dándole una mayor prioridad a su pronta ejecución).

2. *Combinar el producto cartesiano con una operación de selección subsiguiente cuyo predicado represente una condición, para formar una operación de* equi-reunión.

El producto cartesiano es indeseable debido a que su procesamiento resulta demasiado costoso, por ello es recomendable evitar el producto cartesiano y en su lugar utilizar operaciones de equi-reunión.

3. *Utilizar la asociatividad de las operaciones binarias para dar una mayor prioridad a la realización temprana de las operaciones de selección más restrictivas.*

Es recomendable realizar el máximo posible de reducciones antes de ejecutar las operaciones binarias. Por ejemplo, si se tienen que realizar dos operaciones de reunión consecutivas, se pueden utilizar las reglas referentes a la asociatividad de la Equi-reunion (y de la unión e intersección) para reordenar las operaciones de modo que se ejecuten primero las relaciones que generen la menor cantidad de datos, lo que significa que la segunda reunión también estará basada en un primer operando más pequeño.

4. *Realizar las operaciones de proyección lo antes posible.*

Las operaciones de proyección reducen la cardinalidad de las relaciones y, por lo tanto el siguiente procesamiento de las mismas. Con este fin pueden utilizarse las reglas referentes a la conmutatividad de la proyección y de las operaciones binarias con el fin de mover las operaciones de proyección lo más abajo posible en el árbol de consulta.

5. *Calcular una única vez las expresiones comunes.*

En caso de que una expresión aparezca varias veces en el árbol de consulta y si el resultado que produce no es excesivamente grande, entonces resulta conveniente almacenar el resultado después de haberlo calculado la primera vez y luego reutilizarlo cada vez que este se requiera. Cabe aclarar que esto representará una ventaja únicamente si el tamaño del resultado correspondiente a la expresión común es lo suficientemente pequeño como para almacenarlo en memoria principal o como para poder acceder a él en el almacenamiento secundario con un costo inferior al de recalcular el resultado.

3.6.1.3 Método de optimización de consultas basado en costos.

Los optimizadores basados en costos generan una gama de planes de evaluación a partir de la consulta recibida empleando las reglas de equivalencia y escogen el de costo mínimo, en [21] [11] se menciona que la generación de planes de ejecución de consultas basada en costos involucra las siguientes etapas:

1. **Generación de expresiones que son lógicamente equivalentes a la expresión original.**

Para ello es necesario que el módulo optimizador desarrolle expresiones equivalentes de la expresión original, mediante las reglas de equivalencia (ya que estas especifican como se puede transformar una expresión a equivalentes lógicos).

2. **Anotación de las expresiones resultantes en maneras alternativas para producir los planes de evaluación de las consultas**

Posterior a la creación las expresiones equivalentes se crean anotaciones referentes a los planes de evaluación para cada nuevo plan.

3. **Estimación del costo de cada plan de ejecución, eligiendo aquel que posee el menor costo.**

Un optimizador basado en costos explora el espacio de todos los posibles planes que son equivalentes a la consulta y elige aquel que tiene el costo menor, para ello se realiza el cálculo de las estadísticas de los planes de ejecución generados, con ello es posible calcular el costo de cada una de las operaciones involucradas en cada plan. Los costos individuales son combinados para determinar un estimado del costo de evaluar cada expresión de algebra relacional.

Un inconveniente de la optimización basada en costos, es el costo de la misma optimización. A pesar de que el costo de la consulta puede ser reducido mediante algoritmos especializados, el número de los distintos planes para una consulta puede ser muy amplio, y el encontrar el plan óptimo para la consulta puede requerir demasiados recursos de cómputo; por lo tanto los optimizadores suelen utilizar heurísticas para reducir el costo de la optimización. La aproximación heurística utiliza reglas de transformación para convertir la expresión de algebra relacional en una forma equivalente, que sea más eficiente.

Finalmente se elige un plan de ejecución de consulta basada en la estimación del costo de los planes. Debido a que solo se manejan estimaciones de los costos, el plan seleccionado puede no ser el plan óptimo; sin embargo, debe ser al menos aproximado a este.

3.6.1.3.1. Componentes de costo de ejecución de una consulta.

El costo de ejecución de una consulta incluye los siguientes componentes:

1. *Costo de acceso al almacenamiento secundario:* Es el costo de la búsqueda, lectura y escritura de bloques de datos que residen en almacenamiento secundario, principalmente en disco. El costo de la búsqueda de registros en un archivo depende del tipo de las estructuras de acceso a dicho archivo, por ejemplo, el ordenamiento, la dispersión y los índices primarios y secundarios. Aparte de esto, algunos factores, como la ubicación contigua de los bloques del archivo en el mismo cilindro del disco o diseminados por el disco, afectan al costo de acceso.

2. *Costo de almacenamiento:* Es el costo de almacenamiento de los resultados intermedios generados por una estrategia de ejecución de la consulta.

3. *Costo computacional:* Es el costo de la ejecución de operaciones en memoria sobre los búferes de datos durante la ejecución de la consulta. Este tipo de operaciones incluye la búsqueda y la ordenación de los registros, la mezcla de registros durante una concatenación y la ejecución de cálculos sobre valores de los campos.

4. Costo de uso de memoria. Es el costo relativo al número de búferes de memoria que se necesitan durante la ejecución de la consulta.

5. *Costo de comunicaciones:* Es el costo del envío de la consulta y de sus resultados desde el sitio donde se ubica la base de datos hasta el sitio donde se originó la consulta.

En el caso de bases de datos de gran tamaño, el objetivo principal es minimizar el costo de acceso al almacenamiento secundario. Las funciones de costo sencillas ignoran otro tipo de factores y comparan las diferentes estrategias de ejecución en función del número de transferencias de bloque entre el disco y la memoria principal.

En el caso de bases de datos de menor tamaño, en las que se puede almacenar en memoria la mayoría de los datos de los archivos implicados en la consulta, el objetivo principal es minimizar el costo computacional.

3.6.1.3.1.1. Modelo de costos distribuido.

En los sistemas distribuidos se deben considerar varios aspectos, entre los que se incluyen el costo de la transmisión de los datos por la red y la ganancia potencial en rendimiento si se hace que varios sitios procesen en paralelo (siempre y cuando su grado de autonomía lo permita) partes de la consulta, por ello se debe de contar con un modelo de costos distribuido que considere los factores propios de la red de comunicaciones para el desarrollo de planes de ejecución eficientes.

Función de costo.

El costo de una estrategia de ejecución distribuida puede ser expresado como la sumatoria del tiempo que le toma a cada módulo involucrado en el procesamiento de la consulta llevar a cabo su trabajo, en [22] expresa una formula general para determinar el tiempo total puede ser especificada de la siguiente manera:

$$Tiempo_{Total} = T_{CPU} * \#_{Instr} + T_{E/S} * \#_{E/S} + T_{MNS} * \#_{MNS} + T_{TR} * \#Bytes$$

En esta expresión los dos primeros componentes miden el tiempo de procesamiento, donde T_{CPU} es el tiempo de una instrucción del CPU, $T_{E/S}$ es el tiempo de acceso a disco. La comunicación es considerada en los últimos dos componentes T_{MNS} es el número establecido de mensajes para la transmisión y T_{TR} es el tiempo que toma transmitir una unidad de dato de un sitio a otro.

La unidad de dato esta expresada en Bytes (El factor #Bytes es la suma de los tamaños de todos los mensajes) o en algunos caso puede ser puesta en términos de otras unidades como paquetes o bloques. Por practicidad el factor T_{TR} suele considerarse como un valor constante, por lo tanto el tiempo de comunicación involucrado para transmitir #Bytes de un sitio a otro es una función lineal de #Bytes.

$$CT(\#Bytes) = T_{MNS} + *\#Bytes$$

3.6.1.3.2. Estrategia de búsqueda.

Ahora que se cuenta con el contexto en el que se maneja el procesamiento de consultas y en específico de la etapa de optimización, es necesario abordar en qué consisten las diversas estrategias que se utilizan para explorar el espacio de búsqueda, en donde usualmente se utilizan las estrategias siguientes [21]:

- De enumeración.
- Aleatorias.

3.6.1.2.1. Estrategia de enumeración.

Esta estrategia utiliza el principio de programación dinámica, por lo tanto para una determinada consulta se enumeran todos los posibles planes de ejecución, lo cual puede conllevar a utilizar un espacio de búsqueda demasiado extenso en el caso de las consultas complejas.

Los planes de ejecución son construidos a partir de planes parciales que ya se encuentran optimizados, iniciando con todas las relaciones o al menos con una parte principal de la consulta; en las soluciones completamente generadas, únicamente el plan de ejecución óptimo es devuelto para su ejecución.

A pesar de que esta técnica brinda excelentes resultados, su desventaja radica en su complejidad exponencial, lo cual ha hecho que en los últimos años se hayan desarrollado diversos algoritmos para brindar una solución alternativa; sin embargo, para estrategias enumerativas se pueden usar heurísticas para disminuir el espacio de búsqueda.

3.6.1.2.2. Estrategias aleatorizadas.

Dado a que las estrategias de enumeración son inadecuadas para optimizar consultas complejas debido a que su espacio de búsqueda se vuelve rápidamente muy grande, surgen las estrategias aleatorizadas.

Las estrategias aleatorizadas inician generalmente con un plan de ejecución inicial el cual es mejorado iterativamente por la aplicación de un conjunto de reglas de transformación, en donde el plan inicial puede ser obtenido mediante una estrategia enumerativa acondicionada con heurísticas Entre los algoritmos más representativos de este tipo se encuentran: el recocido simulado, la mejora iterativa y la optimización de dos fases, estos son algoritmos genéricos que se pueden aplicar a una amplia gama de problemas de optimización.

Dada una cantidad finita de tiempo, los algoritmos aleatorios tienen un rendimiento que depende de las características de la función de costo y de las condiciones iniciales. A lo largo de la historia estos algoritmos se han estudiado para su utilización en la optimización de consultas, sus resultados han sido comparados contra otros métodos de programación dinámica [23] [24] [25] [26] [27] y pese a que los resultados de estas comparaciones varían en función de factores propios de cada implementación, configuración y de las decisiones tomadas por otros módulos de la optimizador de consultas (el espacio de búsqueda y el modelo de costos), es posible señalar que en las consultas que involucran hasta aproximadamente diez combinaciones, es preferible utilizar la programación dinámica en lugar de los algoritmos aleatorios porque resulta más rápido y está garantizado el encontrar al plan óptimo. Mientras que para consultas con un número mayor a este, la situación se invierte, ya que los algoritmos aleatorios, por lo general tienden a generar un plan aceptable muy rápidamente.

3.6.1.3.3. Evaluación *y* elección de un plan de ejecución.

La optimización de consultas de los SABD tradicionales se basan en varias suposiciones fundamentales para la creación del plan de ejecución:

- Los costos son predecibles: Se asume que es posible estimar el número de accesos a disco que serán necesarios para cargar una relación, índice, etc.
- Los costos son estáticos y el rendimiento del CPU es estimado de una manera precisa.
- La fuente de datos al recibir la consulta dedicará inmediatamente su poder de cómputo en la ejecución de la consulta.

La evaluación de una expresión que contienen múltiples operaciones se realiza evaluando cada una en el orden apropiado, y el resultado de cada evaluación es materializado en una relación temporal para después procesar el resultado de la expresión predecesora. Una desventaja de este procedimiento reside en el hecho de tener que crear relaciones temporales, las cuales posteriormente deben ser almacenadas en disco a menos que sean sumamente pequeñas.

Otra alternativa para llevar a cabo la evaluación sin tener que escribir los resultados de muchas expresiones en disco consiste en evaluar varias operaciones de manera simultánea utilizando un pipeline (canal de comunicación), transmitiendo los resultados a las siguientes operaciones.

3.6.1.4 Método de optimización de consultas alternativas.

La optimización de consultas es un campo en el que se han llevado numerosas investigaciones, habiéndose propuesto diversas alternativas al algoritmo de programación dinámica de *System R*. Por ejemplo la técnica de "templado simulado" (en inglés conocido como *Simulated Annealing*) realiza una búsqueda en un grafo cuyos nodos están constituidos por estrategias de ejecución alternativas (esta técnica modela el proceso de templado mediante el que se pueden aumentar el tamaño de cristales, calentando primero el fluido que los contiene y luego dejándolo enfriarse lentamente). Cada nodo tiene un costo asociado y el objetivo buscado por el algoritmo es localizar un nodo con un costo global mínimo. Un movimiento de un nodo a otro se considerará cuesta abajo (cuesta arriba) si el costo del nodo de origen es superior (inferior) al costo del nodo de destino. Un nodo será un mínimo local si para todas las rutas que dan comienzo en ese nodo cualquier movimiento cuesta abajo viene siempre precedido de un movimiento cuesta arriba. Un nodo será mínimo global si tiene el costo menor de entre todos los nodos. El algoritmo realiza un paseo aleatorio continuo aceptando siempre los movimientos cuesta abajo y aceptando los movimientos cuesta arriba con un cierto valor de probabilidad, para tratar de no quedarse con un mínimo local de alto costo. La probabilidad de aceptar un movimiento cuesta arriba decrece con el tiempo y llega al final a ser cero, en cuyo momento se detiene la búsqueda y se devuelve como estrategia optima de ejecución al nodo visitando que tenga el costo menor [25].

El algoritmo de mejora iterativa realiza una serie de optimizaciones locales, comenzando cada una de ellas de un nodo aleatorio y aceptando repetidamente movimientos cuesta abajo hasta que se alcanza un mínimo local [23].

El algoritmo de optimización en dos fases es un hibrido del templado simulado y de la mejora iterativa. En la primera fase se utiliza la mejora iterativa para realizar determinadas optimizaciones locales que dan como resultado algún mínimo local. Este mínimo local se utiliza como entrada para la segunda fase, que está basada en el templado simulado con una baja probabilidad de partida para los desplazamientos cuesta arriba [26].

Los algoritmos genéticos, que simulan un fenómeno biológico, también se han aplicado a la optimización de consultas. Los algoritmo comienzan con una población inicial, compuesta por un conjunto aleatorio de estrategias extraídas de esa población para generar descendientes que heredan las características de ambos padres, aunque estos descendientes pueden sufrir pequeñas modificaciones de forma aleatoria (mutación). Para la siguiente generación, el algoritmo retiene los padres/hijos de costo menor. El algoritmo termina cuando toda la población está compuesta por copias de la misma estrategia (optima).

El algoritmo heurístico A* se ha utilizado en inteligencia artificial para resolver problemas de búsqueda complejos y también puede se ha empleado en la optimización de consultas [28].

A diferencia del algoritmo de programación dinámica anteriormente comentado, el algoritmo A* genera una estrategia completa mucho antes que la programación dinámica y es capaz de realizar la poda del árbol de búsqueda de manera mucho más agresiva.

3.6.2. Fórmulas estadísticas utilizadas en la optimización de consultas.

Para la creación de un plan de ejecución, es necesario estimar el número de registros que se obtendrá de:

- Las consultas a las fuentes de datos locales.
- La adición de una cláusula de filtrado o de un conjunto de estas.
- Una reunión entre dos fuentes distintas.
- Una reunión entre relaciones intermedias o temporales.
- Los valores distintos obtenidos.

A continuación se mostrará la notación (Véase Tabla 3.3) de las fórmulas estadísticas utilizadas por el prototipo desarrollado para la generación de planes de ejecución.

Tabla 3.3 Notación utilizada en las fórmulas.

Representación	Significado
E	Entidad, relación o tabla.
A	Atributo o campo.
X	Valor de comparación introducido por el usuario.
C	Número de tuplas resultantes.
n_E	Número de tuplas en una tabla E.
t_E	Tamaño de una tupla de E.
V(A, E)	V(A, E): Número de valores distintos que aparecen en la relación E por parte del atributo A.

3.6.2.1. Estimación del número de tuplas resultantes de una condición de filtrado.

Mediante la utilización de las condiciones de filtrado es posible reducir el número de tuplas que son devueltas por una consulta, en estas se especifican los criterios que debe cumplir un registro para ser considerado útil para solucionar la necesidad de información del usuario.

La estimación del número de tuplas resultantes de una condición de filtrado depende del operador utilizado para hacer la comparación, como lo muestran el siguiente apartado (Véase Tabla 3.4):

Tabla 3.4 Estimación número de tuplas resultantes de una cláusula de filtrado.

	Operador	Restricción	Valor de c
(a)	$\sigma_{A=x}(E)$	Ninguna	$n_E / V(A,E)$
(b)	$\sigma_{A\neq x}(E)$	Ninguna	$n_E - (n_E / V(A,E))$
(c)	$\sigma_{A \leq x}(E)$	Si x < min(A,E)	0
		Si x > min(A,E)	$n_E \cdot \frac{x - \min(A,E)}{\max(A,E) - \min(A,E)}$
		En ausencia de información estadística	$n_E / 2$
(d)	$\sigma_{A \geq x}(E)$	Si x > max(A,E)	0
		Si x < max(A,E)	$n_E \cdot \frac{\max(A,E) - x}{\max(A,E) - \min(A,E)}$
		En ausencia de información estadística	$n_E / 2$

Para comprender las anteriores formulas supóngase que se tienen dos conjuntos de elementos que emulan dos atributos en una relación, con las siguientes características (Véase Tabla 3.5):

Tabla 3.5 Características de los conjuntos del ejemplo.

Campo C_1	Campo C_2
C_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	C_2 = [2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]
n_{C1} = 10	n_{C2} = 14
V(C_1, E) = 10	V(C_2, E) = 4
min(C_1,E) = 1	min(C_2,E) = 2
max(C_1,E) = 10	max(C_2,E) = 5

(a) Si se desea saber el número de registros cuyo valor es igual a 4 (Véase Tabla 3.6).

Tabla 3.6 Ejemplo al aplicar el operador igualdad en una condición de filtrado.

De C_1		De C_2	
Valor calculado	**Valor real**	**Valor calculado**	**Valor real**
10 / 10 = 1	1	14 / 4 = 3.5	4

(b) Si se desea saber el número de registros cuyo valor es distinto a 4 (Véase Tabla 3.7).

Tabla 3.7 Ejemplo al aplicar el operador desigualdad en una condición de filtrado.

De C_1		De C_2	
Valor calculado	**Valor real**	**Valor calculado**	**Valor real**
10 – (10 / 10) = 9	9	14 – (14 / 4) = 11.5	10

(c) Si se desea saber el número de registros cuyo valor es menor o igual a 4 (Véase Tabla 3.8).

Tabla 3.8 Ejemplo al aplicar el operador menor que en una condición de filtrado.

De C_1		De C_2	
Valor calculado	**Valor real**	**Valor calculado**	**Valor real**
10 * ((4 – 1) / (10 – 1)) = 3.33	4	14 * ((4 – 2) / (5 – 2)) = 9.24	9

(d) Si se desea saber el número de registros cuyo valor es mayor o igual a 4 (Véase Tabla 3.9).

Tabla 3.9 Ejemplo al aplicar el operador mayor que en una condición de filtrado.

De C_1		De C_2	
Valor calculado	**Valor real**	**Valor calculado**	**Valor real**
10 * ((10 – 4) / (10 – 1)) = 6.66	6	14 * ((5 – 4) / (5 – 2)) = 4.66	5

3.6.2.2. Estimación del número de tuplas resultantes de condiciones de filtrado especiales.

La selectividad de una condición de filtrado θ_i es la probabilidad que una tupla en una relación t logre satisfacer θ_i.

Sea s_i el número de registros resultantes de la condición de filtrado θ_i en la entidad R, entonces la selectividad de θ_i es definida como s_i / n_t.

En caso de contar con una agrupación de cláusulas conjuntivas o disyuntivas, entonces el número de tuplas resultantes se calcula de la siguiente manera (Véase Figura 3.6 y Figura 3.7):

a) **Conjunción:** $\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n}(t)$. *Asumiendo independencia de datos*:

$$n_t * \frac{s_1 * s_2 * \ldots * s_n}{n_t^n}$$

Figura 3.6 Estimación número de tuplas de un varias cláusulas conjuntivas.

b) **Disyunción:** $\sigma_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n}(t)$. Estimación del número de tuplas:

$$n_t * \left(1 - (1 - \frac{s_1}{n_t}) * (1 - \frac{s_2}{n_t}) * \ldots * (1 - \frac{s_n}{n_t})\right)$$

Figura 3.7 Estimación número de tuplas de un varias cláusulas disyuntivas.

Usando la información del ejemplo utilizado para comprobar la estimación del número de tuplas resultantes de una condición de filtrado, se ilustrará como se aplica en un conjunto de estas:

(a) Si se desea saber el número de registros cuyo valor es mayor o igual a 2, y menor o igual que 5 (Véase Tabla 3.10).

Tabla 3.10 Ejemplo al utilizar predicados conjuntivos.

De C_1		De C_2	
Valor calculado	**Valor real**	**Valor calculado**	**Valor real**
$S_1 = 10 * ((5 - 1) / (10 - 1)) = 4.44$ $S_2 = 10 * ((10 - 2) / (10 - 1)) = 8.88$ $(10*(S_1 * S_2))/(10^2) = 3.94$	4	$S_1 = 14 * ((5 - 2) / (5 - 2)) = 14$ $S_2 = 14 * ((5 - 2) / (5 - 2)) = 14$ $(14*(S_1 * S_2))/(14^2) = 14$	14

(b) Si se desea saber el número de registros cuyo valor es mayor o igual a 2, o menor o igual que 5 (Véase Tabla 3.11).

Tabla 3.11 Ejemplo al utilizar predicados disyuntivos.

De C_1		De C_2	
Valor calculado	**Valor real**	**Valor calculado**	**Valor real**
$S_1 = 10 * ((5 - 1) / (10 - 1)) = 4.44$ $S_2 = 10 * ((10 - 2) / (10 - 1)) = 8.88$ $10*(1-(1-(S1/10))* (1-(S2/10))) = 9.37$	10	$S_1 = 14 * ((5 - 2) / (5 - 2)) = 14$ $S_2 = 14 * ((5 - 2) / (5 - 2)) = 14$ $14*(1-(1-(S1/14))* (1-(S2/14))) = 14$	14

3.6.2.3. Estimación del número de tuplas resultantes de una reunión.

Para realizar el cálculo del número de tuplas resultantes de la operación de reunión, deben considerarse algunos aspectos:

Sean R y S dos esquemas de relaciones distintas y sean r y s las instancias de dichos esquemas.

- Si $R \cap S = \varnothing$, es decir, si ambas relaciones no poseen atributos en común, entonces el número estimado de tuplas resultantes de la operación Join(R, S) puede ser igual al producto cartesiano:

$$c = |R| \times |S|.$$

- Si $R \cap S = \{A\}$, de modo que A (un campo o un conjunto de ellos) no es una llave primaria para *R* o *S* y se supone que todos los valores aparecen con igual probabilidad, cada tupla en R produciría tuplas en Join(R, S).

  Bajo este contexto, en el caso de una tupla t que pertenezca a R, y suponiendo que existe A (un campo o un conjunto de ellos, cuyo dominio exista en R y S), se estima que el Join(R, S), produce un número de tuplas igual a:

$$\frac{n_r * n_s}{V(A,r)}$$

  Considerando todas las tuplas de R, se estima que produce un número de tuplas igual a

$$\frac{n_s}{V(A,s)}$$

  Invirtiendo los papeles ente R y de S en la estimación anterior, se obtiene la estimación de:

$$\frac{n_r * n_s}{V(A,s)}$$

  En la literatura se señala que el menor valor de estas dos estimaciones suele ser el más preciso (Véase Figura 3.8):

$$c = \min((\frac{n_r * n_s}{V(A,r)}), (\frac{n_r * n_s}{V(A,s)}))$$

Figura 3.8 Estimación número de tuplas de una reunión.

3.6.2.4. Estimación del número de valores distintos en una reunión.

Para las selecciones el número de valores distintos de un atributo (o de un conjunto de atributos) A en el resultado de una selección V (A, σ_θ (R)), puede estimarse de las maneras siguientes:

- Si la condición de selección θ obliga a que *A* adopte un valor en específico, por ejemplo, *A* = 3, $V(A, \sigma_\theta(R)) = 1$.

- Si θ obliga a que *A* adopte un valor de entre un conjunto especificado de valores (por ejemplo, (*A* = 12 *A* = 32 *A* = 4)), entonces *V* (*A,* σ_θ (*R*)) se define como el número de valores especificados.

- Si la condición de selección θ es de la forma *A op v*, donde *op* es un operador de comparación, *V* (*A,* σ_θ (*R*)) se estima que es *V* (*A, R*) * *s*, donde *s* es la selectividad de la selección

- En todos los demás casos de selecciones se utiliza una estimación aproximada de min (*V* (*A, R*), $n\sigma_\theta$ (*R*))

- Para las reuniones el número de valores distintos de un atributo (o de un conjunto de atributos) A en el resultado de una reunión, V (A, Join(R,S)), puede estimarse de las maneras siguientes:

- Si todos los atributos de *A* proceden de *R*, *V* (*A,* Join(R,A)) se estima su valor como:

$$Min\ (V\ (A, R), n_{Join\ (R, S)})$$

- En caso contrario si todos los atributos de A proceden de S, entonces el número de valores distintos se calcula como (Véase Figura 3.9):

$$Min\ (V\ (A, S), n_{Join\ (R, S)})$$

Figura 3.9 Estimación del número de valores distintos.

3.6.2.5. Histogramas.

Algunas bases de datos almacenan la distribución de los valores de cada atributo en forma de histograma (Véase Figura 3.10); en los histogramas los valores del atributo se dividen en una serie de rangos y con cada rango el histograma asocia el número de tuplas cuyo valor del atributo se halla en ese rango. Por ejemplo: el rango de valores del atributo edad de la relación persona puede dividirse en 0-9, 10-19,. . ., 90-99 (suponiendo una edad máxima de 99). Con cada rango se almacena un recuento del número de tuplas persona cuyos valores de edad se hallan en ese rango. Sin la información del histograma un optimizador tendría que suponer que la distribución de los valores es uniforme; es decir, que cada rango tiene el mismo recuento.

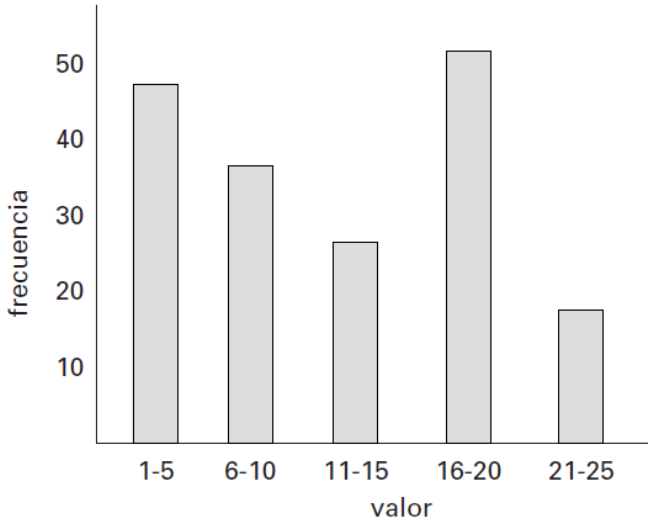


Figura 3.10 Ejemplo de histograma.

Capítulo 4 Análisis y diseño.

4.1. Introducción.

En este capítulo se mostrará el análisis y el diseño utilizado para la implementación del enfoque propuesto para la generación de planes de ejecución, que buscará ser incorporada en el módulo optimizador de consultas globales de un sistema mediador. Por claridad conceptual se utilizarán algunos elementos basados en el lenguaje de modelado unificado (*Unified Modeling Language UML*): casos de uso, diagramas de interacción y diagramas de actividades.

A continuación se detallará sobre el entorno en el que opera un sistema mediador (Véase Figura 4.1):

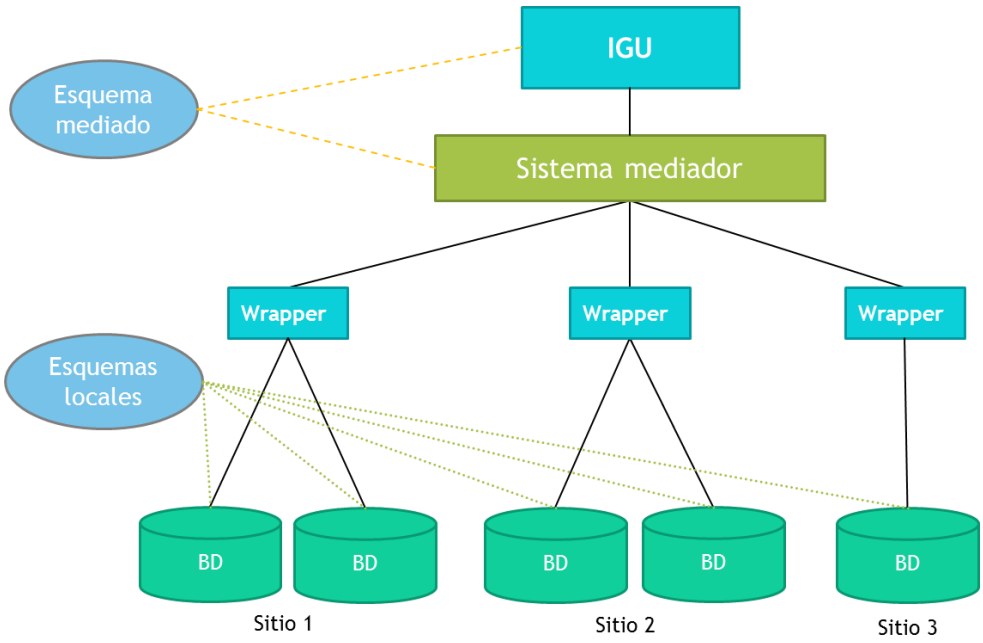


Figura 4.1 Entorno de operación de un sistema mediador.

El sistema mediador interactúa directamente con dos elementos: la Interfaz gráfica de usuario y las fuentes de datos locales, ambos son considerados como actores externos.

- **Interfaz gráfica de usuario (IGU):** Muestra el esquema mediado (contiene los campos de información disponibles mediante la conciliación de los datos contenidos en las distintas fuentes de datos), permite la formulación de consultas basadas en dicho esquema y se encarga de transferirlas al sistema mediador; después de que este resuelve la consulta, la IGU recibe el resultado y lo muestra ante el usuario.

- **Fuentes de datos locales:** El sistema mediador no almacena de modo físico los datos, sino que realiza un conjunto de consultas a las fuentes de datos locales, con el fin de extraer la información necesaria para dar respuesta a la necesidad de información del usuario.

Los sistemas mediadores suelen incorporar adaptadores para lidiar con la heterogeneidad de los modelos de datos contenidos en las fuentes de datos; sin embargo, en el desarrollo se considera únicamente fuentes que usan al modelo relacional.

Para que un sistema mediador sea capaz de llevar a cabo el procesamiento de consultas, emplea los módulos que a continuación se mencionan (Véase Figura 4.2):

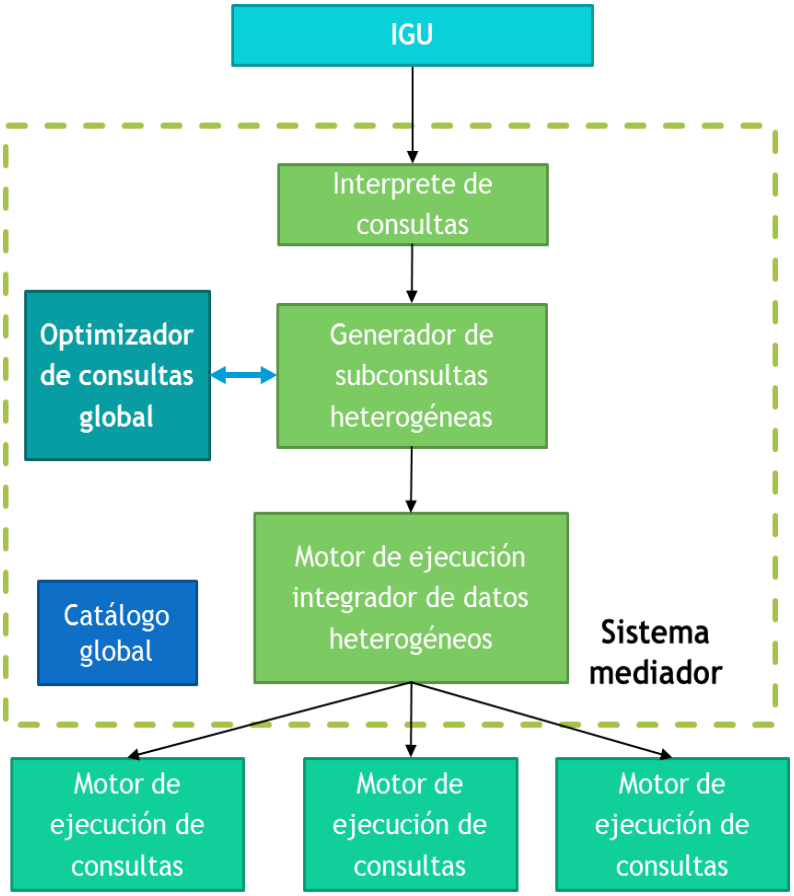


Figura 4.2 Módulos del sistema mediador.

- **La interfaz gráfica de usuario (IGU):** Esta sirve como medio de interacción entre el sistema mediador y el usuario, permitiéndole expresar sus consultas y visualizar los resultados.

- **Intérprete de consultas:** Este componente recibe la consulta de la IGU y la transforma en una representación canónica, que permita una libre manipulación por los demás módulos del sistema mediador, en donde suele utilizarse la representación de grafo o de árbol.

- **Generador de subconsultas heterogéneas:** Si el sistema mediador realiza la integración de información proveniente de fuentes con modelos de datos distintos, entonces necesita de este módulo, para que se encargue de descomponer la consulta mediada en expresiones basadas en los modelos de datos involucrados.

- **Optimizador de consultas global:** Se encarga de generar un plan de ejecución, en el que debe de especificarse la forma en la que se resolverá la consulta, buscando que el proceso de integración se realice de la forma más eficiente posible.

- **Motor de ejecución integrador de datos heterogéneos:** Lleva a cabo la realización del plan de ejecución, y se ocupa de coordinar e interactuar con las fuentes de datos, así como de combinar los resultados intermedios.

- **Motores de ejecución de consultas de las fuentes de datos locales:** Estos se encargan de procesar localmente las consultas realizadas por el sistema mediador.

- **Catálogo del sistema mediador:** Este elemento es fundamental para todos los módulos previamente mencionados, dado que el catálogo contiene la información descriptiva de las fuentes de datos: los esquemas, los metadatos y valores estadísticos que describen la distribución de los valores de los datos.

4.2. Requerimientos funcionales y no funcionales.

En este apartado se detallarán los requerimientos funcionales y no funcionales del módulo optimizador de consultas globales, como responsable de la generación de los planes de ejecución del sistema mediador.

El prototipo deberá cubrir las funcionalidades que se plantean en seguida para que pueda ser incorporado en un sistema mediador.

Requerimientos funcionales del módulo optimizador de consultas globales.

Antes de tiempo de consulta.

- El prototipo deberá disponer de acceso a las fuentes de datos remotas, para crear un catálogo (en un archivo XML, en los apéndices se anexa un ejemplo de su estructura) por cada una de ellas, mediante la extracción de metadatos y la obtención de las estadísticas de la base de datos, como son el número de tuplas de cada tabla, el número de valores distintos y selectividad de cada atributo, así como sus valores máximo y mínimo.

- El prototipo deberá recibir o definir las correspondencias que indicarán los elementos que son afines entre sí hasta cierto umbral; es decir, los atributos que comparten un mismo dominio, a pesar de pertenecer a fuentes de datos distintas.

- El prototipo deberá recibir o definir el esquema mediado, y deberá contener los atributos sobre los que es posible consultar al sistema; además tendrá que incluir los mapeos entre los atributos mediados con respecto a los esquemas de las fuentes de datos remotas.

En tiempo de consulta.

- El prototipo recibe o define la consulta del usuario.

- El prototipo genera como resultado un plan de ejecución, que estará conformado por un conjunto de expresiones para cada una de las fuentes de datos involucradas en la consulta del usuario y por un orden de ejecución específico para dichas expresiones.

Requerimientos no funcionales del módulo optimizador de consultas globales:

- ***Eficiencia****:* Tanto las rutinas involucradas en la generación de planes de ejecución, así como el resultado de estas, vigilarán que el proceso de integración se desarrolle ágilmente buscando reducir el tiempo de espera del usuario por la respuesta a su consulta.

- ***Extensibilidad****:* La implementación desarrollada deberá ser capaz de incorporar nuevos elementos, es decir, deberá permitir que determinadas modificaciones sean implementadas.

- ***Modularidad****:* El prototipo deberá ser independiente con respecto a los demás módulos, tanto como lo permita su propia funcionalidad, con el fin de facilitar la claridad y conceptualización de cada componente, así como para el mantenimiento y la identificación de fallas.

- ***Interoperabilidad**:* La implementación desarrollada deberá de contar con la capacidad de interactuar con otros componentes.

- ***Compatibilidad**:* El prototipo deberá ser capaz de funcionar correctamente con sistemas de bases de datos relacionales (ejemplo: Oracle, MySQL, DB2 y SQL Server) y plataformas distintas.

4.3. Roles de usuario.

El prototipo desarrollado considera dos roles para los usuarios:

- **Usuario común:** Es aquel que expresa mediante una consulta su necesidad de información ante el sistema y visualizar el resultado de su ejecución.

- **Administrador:** Puede desempeñar las funciones de un usuario común, pero debe de poseer información de las fuentes de datos así como acceso a estas, para llevar a cabo la evaluación de las correspondencias generadas de manera semiautomática, complementarlas manualmente y definir el esquema mediado.

4.4. Casos de uso de los requerimientos funcionales de la aplicación.

A continuación, se presentarán los casos de uso de los requerimientos funcionales involucrados con el procesamiento de consultas en un sistema mediador (Véase Figura 4.3).

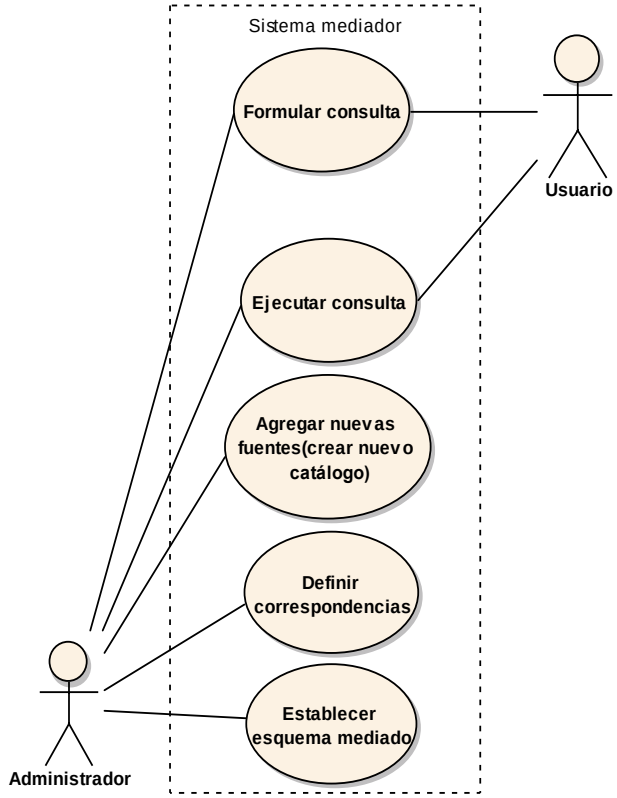


Figura 4.3 Diagrama de los casos de uso existentes en un sistema mediador.

4.4.1. Agregar nuevas fuentes de datos.

Inicialmente deben de definirse las fuentes a las que el sistema mediador tiene acceso, mediante la creación de un catálogo por cada base de datos, en donde se almacena la información referente a los esquemas y valores estadísticos (cardinalidad, selectividad y número de tuplas) que describen la distribución de los datos. Usando estos catálogos el mediador puede conocer los elementos contenidos en cada base de datos.

El catálogo del mediador debe almacenar la siguiente información relevante: la dirección URL de la fuente de datos, el controlador de conexión del SABD, el nombre de usuario y su respectiva contraseña; con estos datos, se realiza una prueba de conexión, se establece la conexión y se procede a extraer la información necesaria desde las fuentes locales, que es almacenada localmente (en un archivo XML, en los apéndices se da un ejemplo de este archivo). Este proceso

se sintetiza en el caso de caso de uso y los diagramas de secuencia y de actividades mostrados a continuación (Véase Tabla 4.1, Figura 4.4, Figura 4.5).

Tabla 4.1 Caso de uso: Agregar nuevas fuentes de datos.

Actores participantes	**Administrador:** Individuo que posee acceso a las fuentes de información, que se desean agregar al sistema.
Condiciones iniciales	• El administrador debe contar con la dirección URL del servidor de la base de datos, el controlador de conexión, el nombre de usuario y la contraseña de la base de datos que desee agregar.
Flujo de eventos	1. El administrador accede a la pantalla del sistema y elige el menú dedicado a la creación de nuevos catálogos. 2. El administrador introduce la información particular de la conexión a la base de datos: la dirección URL, el controlador de conexión, el nombre de usuario y la contraseña de la fuente que desee incorporar al sistema. 3. El administrador prueba el estado de conexión con los datos provistos. 4. El administrador inicia la generación de los catálogos, la aplicación se conecta a las fuentes de datos y extrae los metadatos, así como los valores estadísticos. 5. Se almacena la información obtenida en un archivo XML.
Condiciones de salida	El sistema obtiene un archivo XML que representa el catálogo por cada fuente de datos.

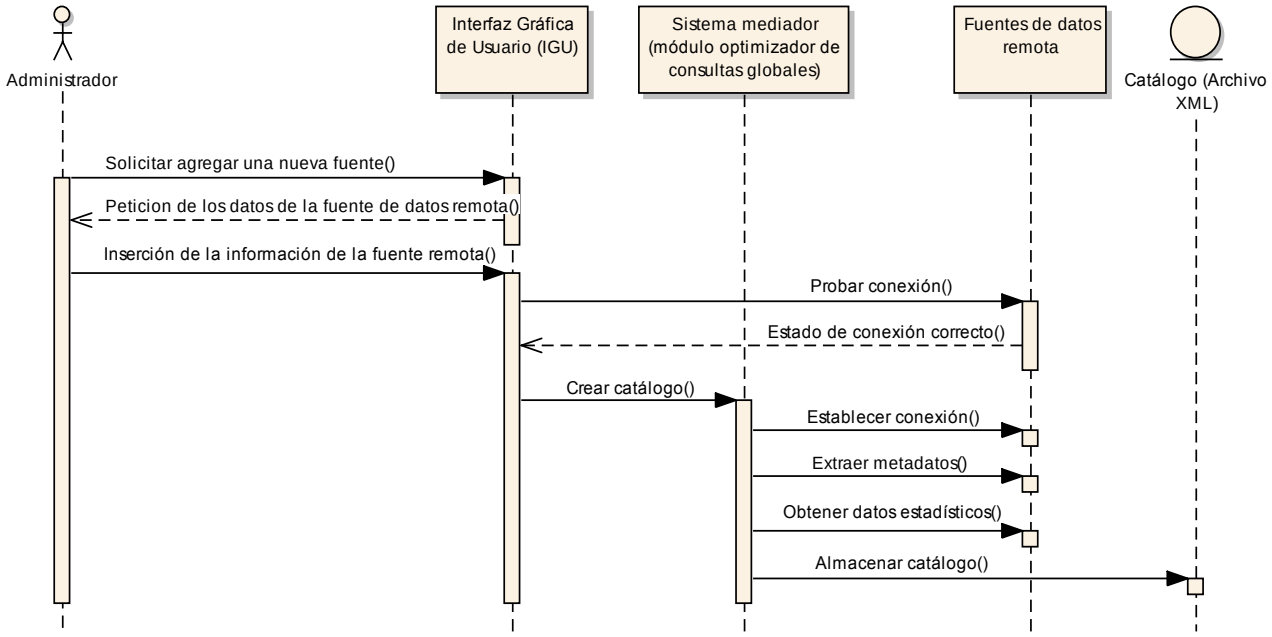


Figura 4.4. Diagrama de secuencia: Agregar nuevas fuentes de datos.

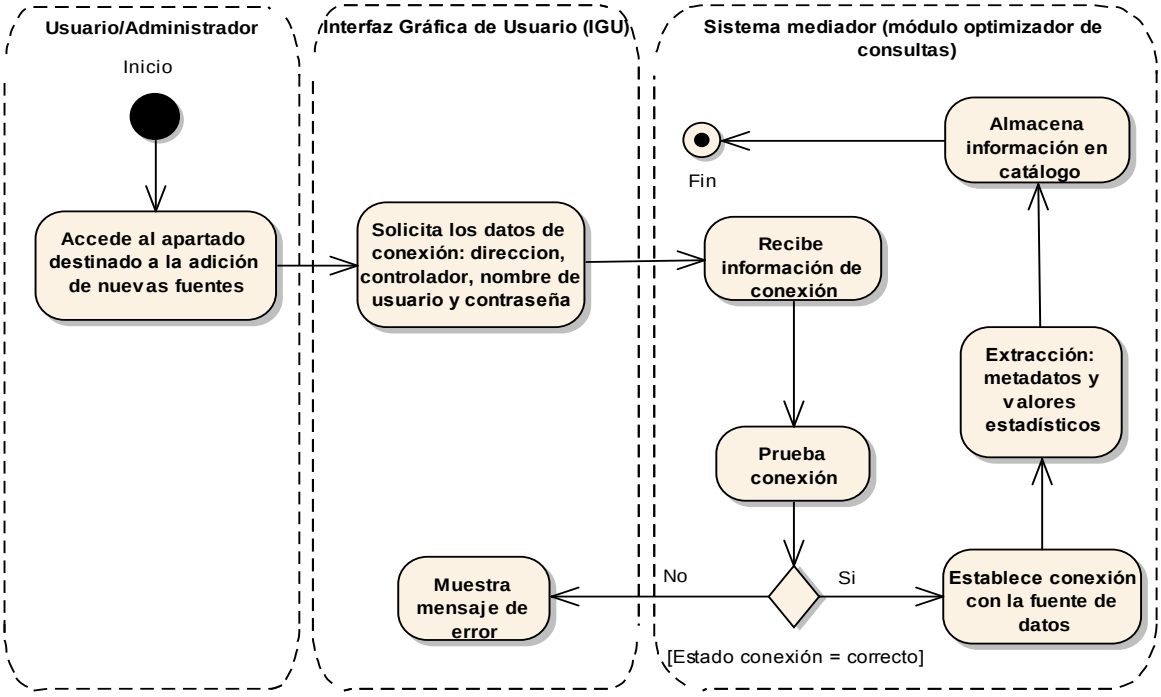


Figura 4.5 Diagrama de actividades: Agregar nuevas fuentes de datos.

4.4.2. Definición de las correspondencias.

Después de que se han creado los catálogos de las fuentes remotas, es necesario que el sistema ubique los elementos que comparten un mismo dominio, por ello se realiza un proceso de tres etapas: primero se ejecuta un algoritmo que realiza un análisis de similitud basándose en los metadatos de los catálogos, después el administrador analiza las correspondencias sugeridas y las complementa de modo manual en caso de que no esté de acuerdo con alguna correspondencia encontrada; una vez que este decide que están completas, ejecuta un algoritmo que aplica la propiedad de transitividad en las correspondencias ya definidas, con el fin de establecer el espacio de combinaciones posibles entre atributos; por ejemplo si existen las correspondencias c_1(A, B) y c_2(B, C) donde A, B y C son fuentes de datos distintas, entonces se crea c_3(A, C) y se incluye como posible combinación. Después que se han definido todas las correspondencias, estas se almacenan en un documento XML para posteriormente ser analizadas por los procesos posteriores. A continuación se resume este proceso en el siguiente caso de uso y los diagramas de secuencia y de actividades (Véase Tabla 4.2, Figura 4.6, Figura 4.7).

Tabla 4.2 Caso de uso: Definición de las correspondencias.

Actores participantes	**Administrador:** Individuo que conoce y domina la información contenida en las fuentes de datos remotas.
Condiciones iniciales	• El sistema mediador valida que existan al menos dos catálogos, lo que implica que este tiene acceso a dos fuentes de datos. • El administrador debe definir un valor de umbral, que determine el nivel de escrupulosidad en el proceso de evaluación de posibles correspondencias.
Flujo de eventos	1. El administrador accede al apartado dedicado a la creación de correspondencias. 2. El administrador selecciona la opción de generar automáticamente un conjunto de posibles correspondencias y define el umbral de similitud que se usará para ello. o Se realiza una comparación de todos los elementos de una fuente de datos con todos los elementos de las demás fuentes de datos. o En caso de que el valor de similitud provisto por la comparación sea mayor o igual al valor de umbral, entonces se almacena provisionalmente la correspondencia.

	3. El administrador analiza las correspondencias generadas de modo semiautomático y decide si estas son correctas; en caso de no serlas las descarta. 4. El administrador termina de definir todas las correspondencias, entonces mediante un algoritmo aplica la propiedad de transitividad y completa las correspondencias faltantes. 5. Se almacenan el conjunto de correspondencias generadas.
Condiciones de salida	El sistema obtiene un conjunto de correspondencias que indican los elementos que tienen cierta afinidad entre las distintas fuentes de datos.

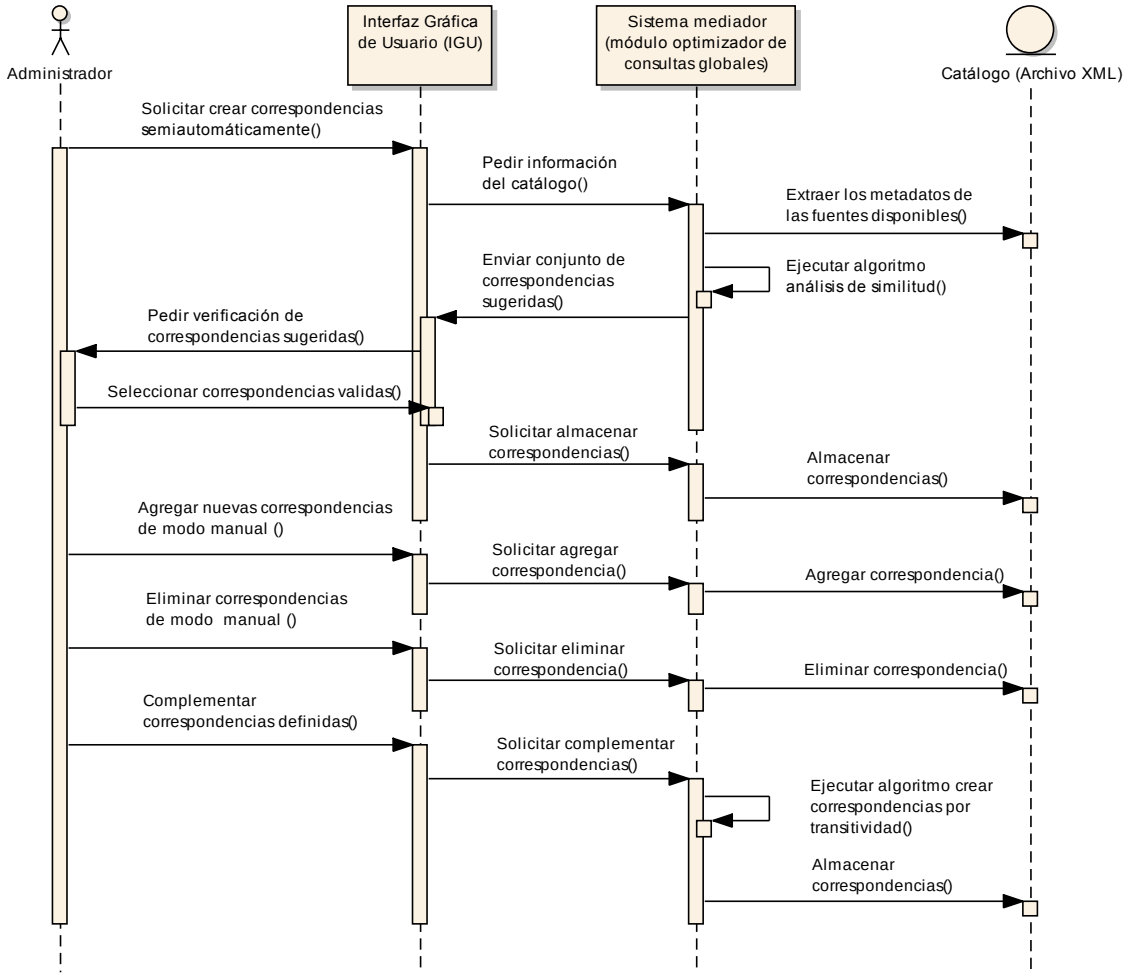


Figura 4.6 Diagrama de secuencia: Definición de las correspondencias.

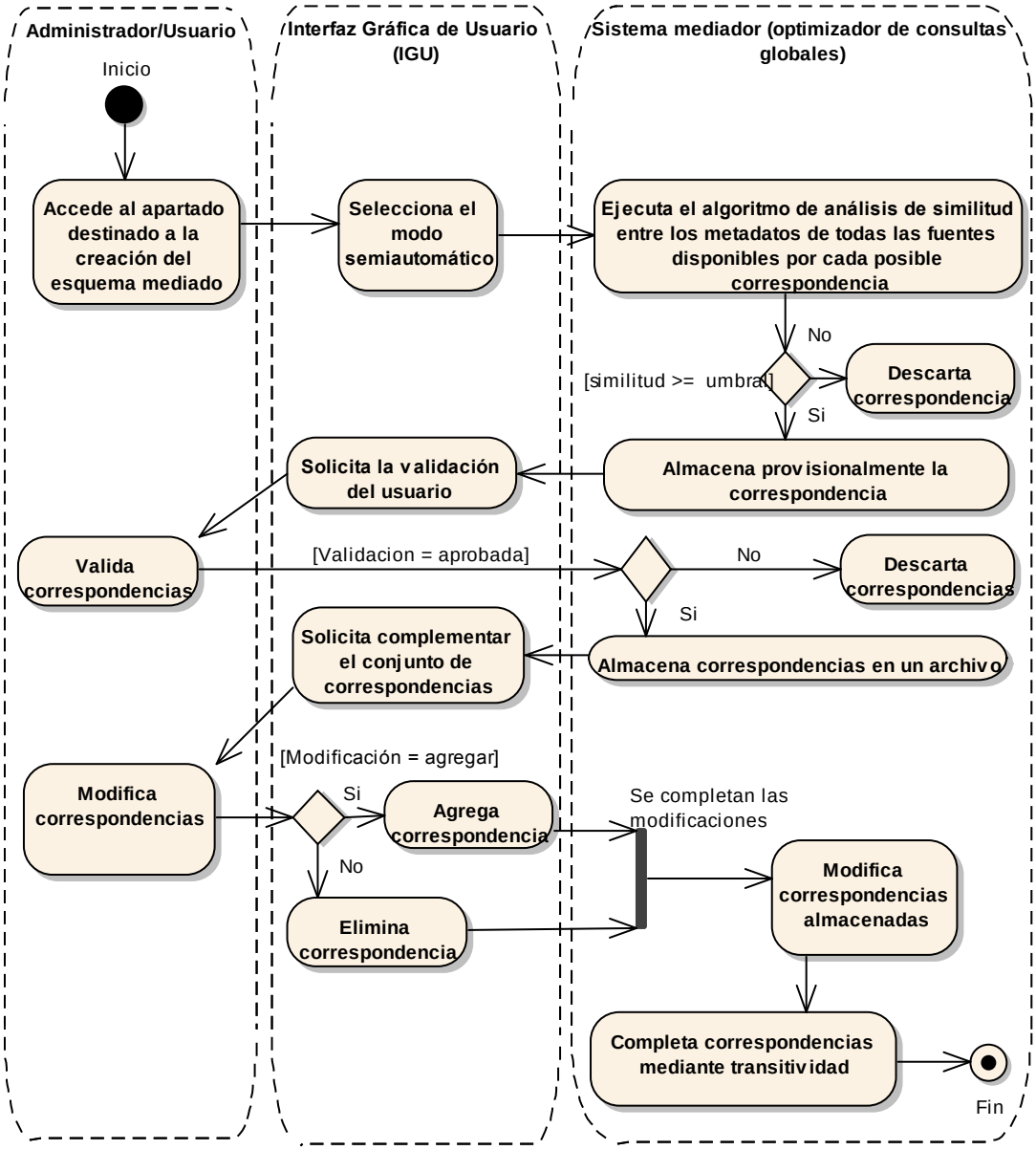


Figura 4.7 Diagrama de actividades: Definición de las correspondencias.

4.4.3. Establecer el esquema mediado.

Una vez que se tienen los catálogos de las fuentes de datos remotas y las correspondencias que señalan los campos que son afines, entonces se procede a definir el esquema mediado, el cual contiene la información que podrá ser consultada mediante el uso del sistema mediador, además de almacenar la correlación entre los atributos mediados y su ubicación física.

Para la definición del esquema mediado, el administrador tendrá que generar un atributo virtual a la vez, proporcionando el nombre de la entidad virtual(objeto utilizado para la agrupación de varios de estos campos), el nombre del atributo, y especificar los campos de los esquemas locales que comparten el mismo dominio, como se sintetiza en el siguiente caso de uso y en los correspondientes diagramas de secuencia y de actividades (Véase Tabla 4.3, Figura 4.8, Figura 4.9).

Tabla 4.3 Caso de uso: Establecer el esquema mediado.

Actores participantes	**Administrador:** Individuo que conoce la información contenida en las fuentes de datos remotas y tiene presente la necesidad de información que se busca satisfacer mediante la utilización del mediador.
Condiciones iniciales	• El sistema mediador valida los catálogos de cada fuente local involucrada. • El sistema mediador revisa conjunto de correspondencias del archivo generado en el proceso anterior, que señalen los elementos que manejan un mismo dominio.
Flujo de eventos	1. El administrador accede al apartado dedicado a la creación del esquema mediado. 2. Por cada atributo que se desea incorporar, el administrador debe especificar: o El nombre de la relación virtual en la que se va a ubicar el nuevo atributo (en caso de que esta no exista, entonces se crea una nueva). o El nombre del atributo (en caso de que este no exista, entonces se crea uno nuevo). o Los campos de las fuentes de datos remotas que abastecerán la información de dicho atributo. 3. Se almacena la información del esquema mediado en un documento XML.
Condiciones	El sistema obtiene la definición del esquema para mantener la relación con las

de salida	fuentes remotas.

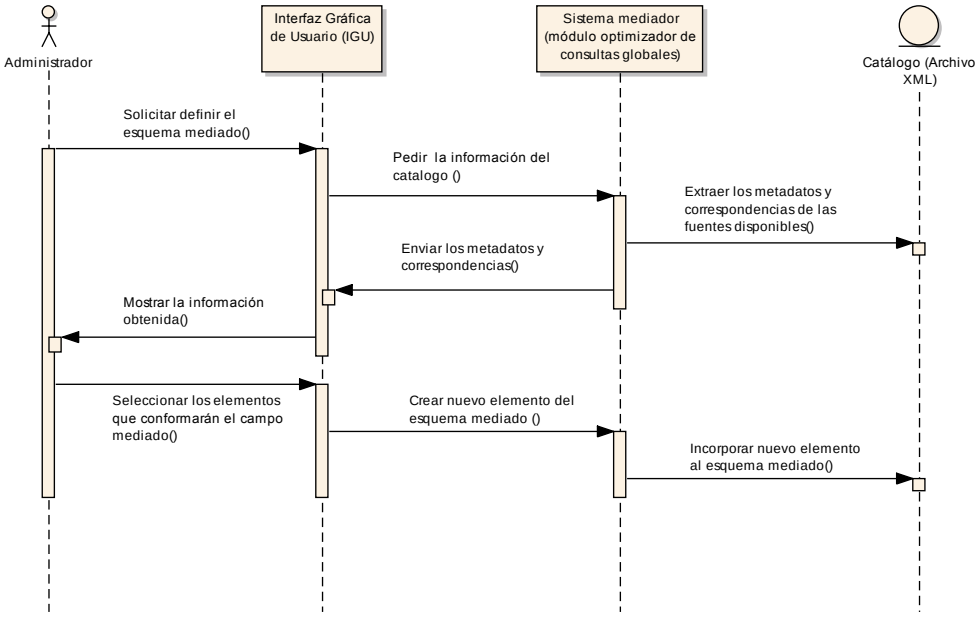


Figura 4.8 Diagrama de secuencia: Establecer el esquema mediado.

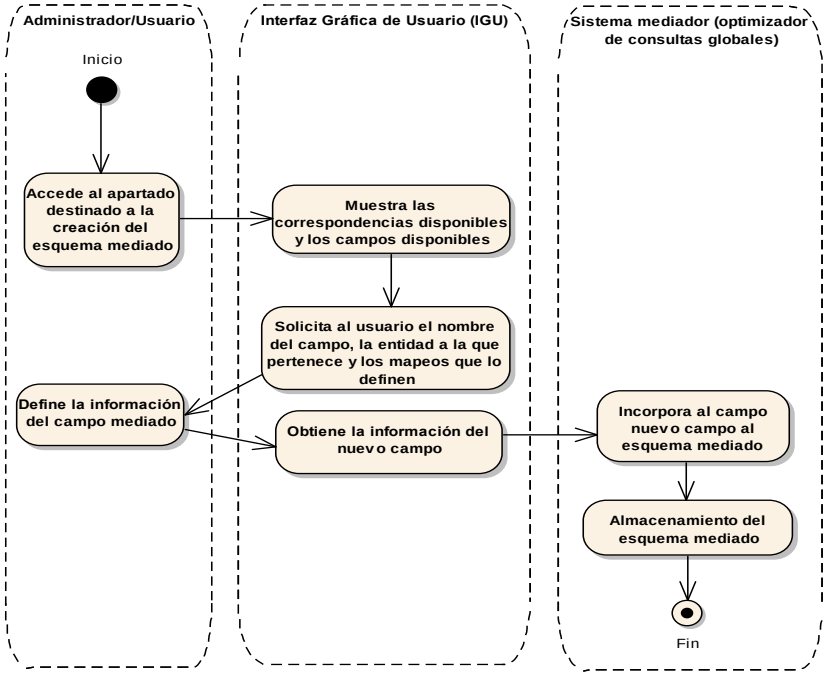


Figura 4.9 Diagrama de actividades: Establecer el esquema mediado.

4.4.4. Formular consulta.

El sistema mediador puede recibir consultas solo después de que se llevó a cabo la creación de los catálogos, de las correspondencias y del esquema mediado; posteriormente el usuario se auxilia de una interfaz gráfica de usuario (IGU) para realizar la formulación de sus consultas. Esta interfaz le permite, de un modo visual e intuitivo, seleccionar los elementos del esquema mediado que desea analizar y especificar las condiciones que deben de cumplirse para que un registro sea de su interés; este proceso se ilustra mediante el siguiente caso de uso y los diagramas de secuencia y de actividades (Véase Tabla 4.4, Figura 4.10, Figura 4.11).

Tabla 4.4 Caso de uso: Formular consulta.

Actores participantes	**Administrador o usuario:** Individuo que desea expresar su necesidad de información ante el sistema.
Condiciones iniciales	• El sistema mediador debe de tener los catálogos de las fuentes de datos del esquema mediado. • El conjunto de correspondencias, que señalen los elementos que manejan un mismo dominio. • La definición del esquema mediado.
Flujo de eventos	1. El usuario visualiza los atributos contenidos en el esquema mediado, mediante la interfaz gráfica de usuario. 2. El usuario determina los atributos que desea visualizar del esquema mediado. 3. El usuario establece las condiciones de filtrado, que determinan las condiciones que deben cumplir los registros.
Condiciones de salida	Se obtiene la consulta (en términos del esquema mediado) del usuario. **En esta aplicación no se considera el componente "Interprete de consultas" debido a que la IGU implementada expresa la consulta en la representación que se utilizará para desarrollar el plan de ejecución.*

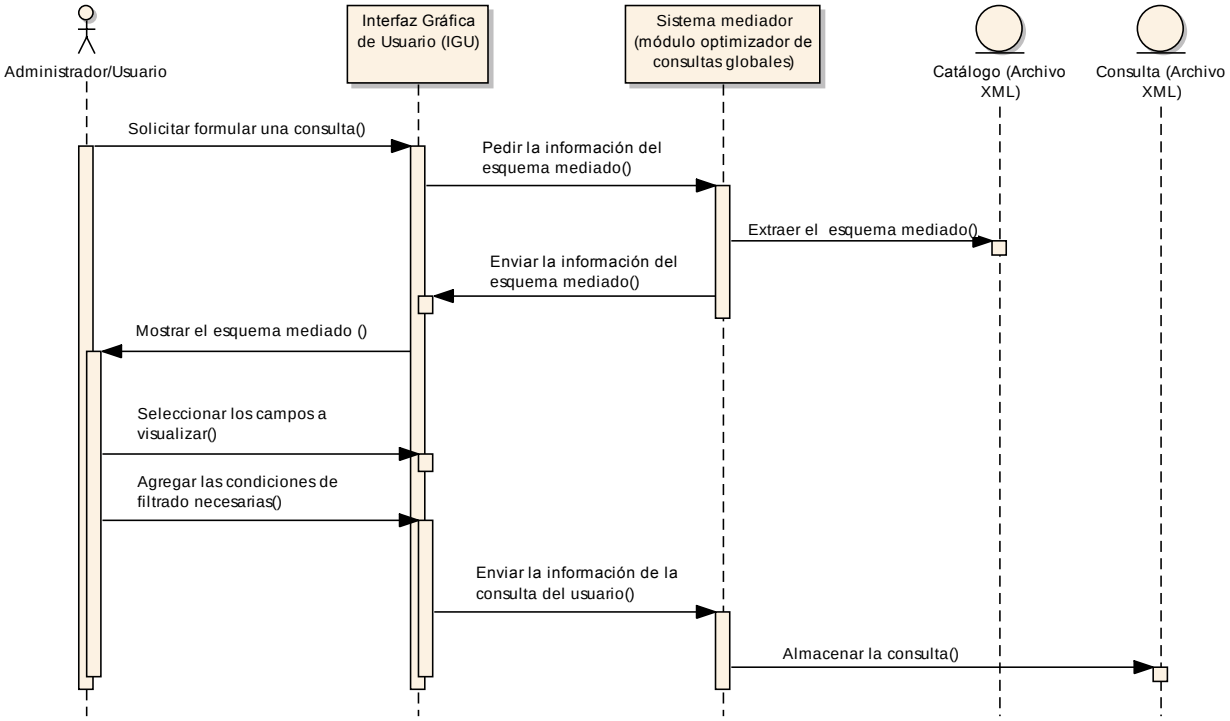


Figura 4.10 Diagrama de secuencia: Formular consulta.

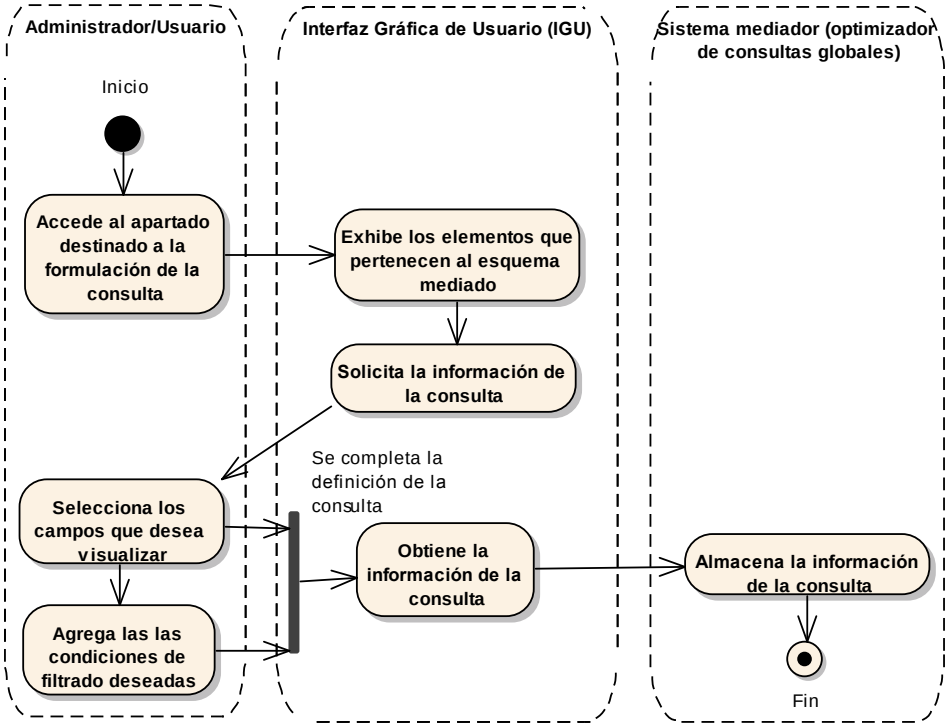


Figura 4.11 Diagrama de actividades: Formular consulta.

4.4.5. Ejecución de la consulta mediada.

Si el sistema mediador está habilitado para procesar información ubicada en fuentes que emplean modelos de datos diferentes, entonces el primer módulo involucrado en el procesamiento de la consulta es el "Generador de subconsultas heterogéneas", que se encarga de descomponer la consulta en expresiones basadas en los modelos de datos implicados y después enviarlas al módulo "Optimizador de consultas globales".

El "Optimizador de consultas globales", recibe la consulta y auxiliándose del catálogo de las fuentes de datos remotas, se encarga de generar expresiones (consultas objetivo) en términos de las fuentes de datos locales; además define en el orden en el que los resultados de estas serán procesados por el integrador del sistema mediador, dicho orden está orientado a mejorar el rendimiento del procesamiento de la consulta. Una vez que dicho plan se ha definido, es transmitido al "Motor de ejecución integrador de datos heterogéneos" (MEIDH).

El MEIDH se encarga de realizar el plan de ejecución global, estableciendo primeramente comunicación con las fuentes de datos remotas extrayendo su información, para después combinar los datos, y finalmente enviar los resultados a la IGU.

El proceso anteriormente expuesto es representado en forma de caso de uso, diagrama de secuencia y diagrama de actividades (Véase Tabla 4.5, Figura 4.12, Figura 4.13).

Tabla 4.5 Caso de uso: Ejecución de la consulta mediada.

Actores participantes	**Administrador o usuario:** Individuo que expresa su necesidad de información ante el sistema mediador. **Módulos del sistema mediador:** ○ Interfaz gráfica de usuario (IGU). ○ Generador de subconsultas heterogéneas (GCH). ○ Optimizador de consultas (OC). ○ Motor de ejecución integrador de datos heterogéneos (MEIDH). Motores de ejecución de consultas de las fuentes de datos locales.
Condiciones iniciales	• Conjunto de correspondencias, que señalen los elementos que manejan un mismo dominio. • La definición del esquema mediado. • La consulta del usuario.
Flujo de eventos	1. El sistema mediador recibe la indicación de procesar la consulta del usuario.

	2. Si el sistema mediador no permite la integración de información almacenada en fuentes con modelo de datos distintos, entonces la consulta es enviada directamente al módulo OC, en caso de contrario la envía al módulo GSH, que se encarga de descomponer la consulta en expresiones para cada modelo de datos involucrado, y posteriormente transmite sus resultados al módulo OC. 3. El módulo OC recibe la consulta y se encarga de descomponerla en consultas objetivo (auxiliándose de la información contenida en los catálogos), que pueden ser contestadas por las fuentes de datos remotas; posteriormente mediante la ejecución de un algoritmo Greedy (que da prioridad a la ejecución temprana de las operaciones de reunión que proveen de una menor cantidad de tuplas) se determina el orden de ejecución, que buscará brindar eficiencia al proceso de integración, y finalmente el plan generado se transmite al módulo de MEIDH. 4. El módulo MEIDH recibe el plan de ejecución y envía las consultas objetivo a las fuentes de datos implicadas en la consulta, extrae los resultados y los combina, finalmente envía el resultado a la IGU. 5. Los “Motores de ejecución de consultas de las fuentes de datos locales” se encargan de procesar las consultas proporcionadas por el sistema mediador y envían los resultados de regreso. 6. La IGU despliega el resultado de la consulta.
Condiciones de salida	El usuario es capaz de visualizar los resultados de la consulta que realizó.

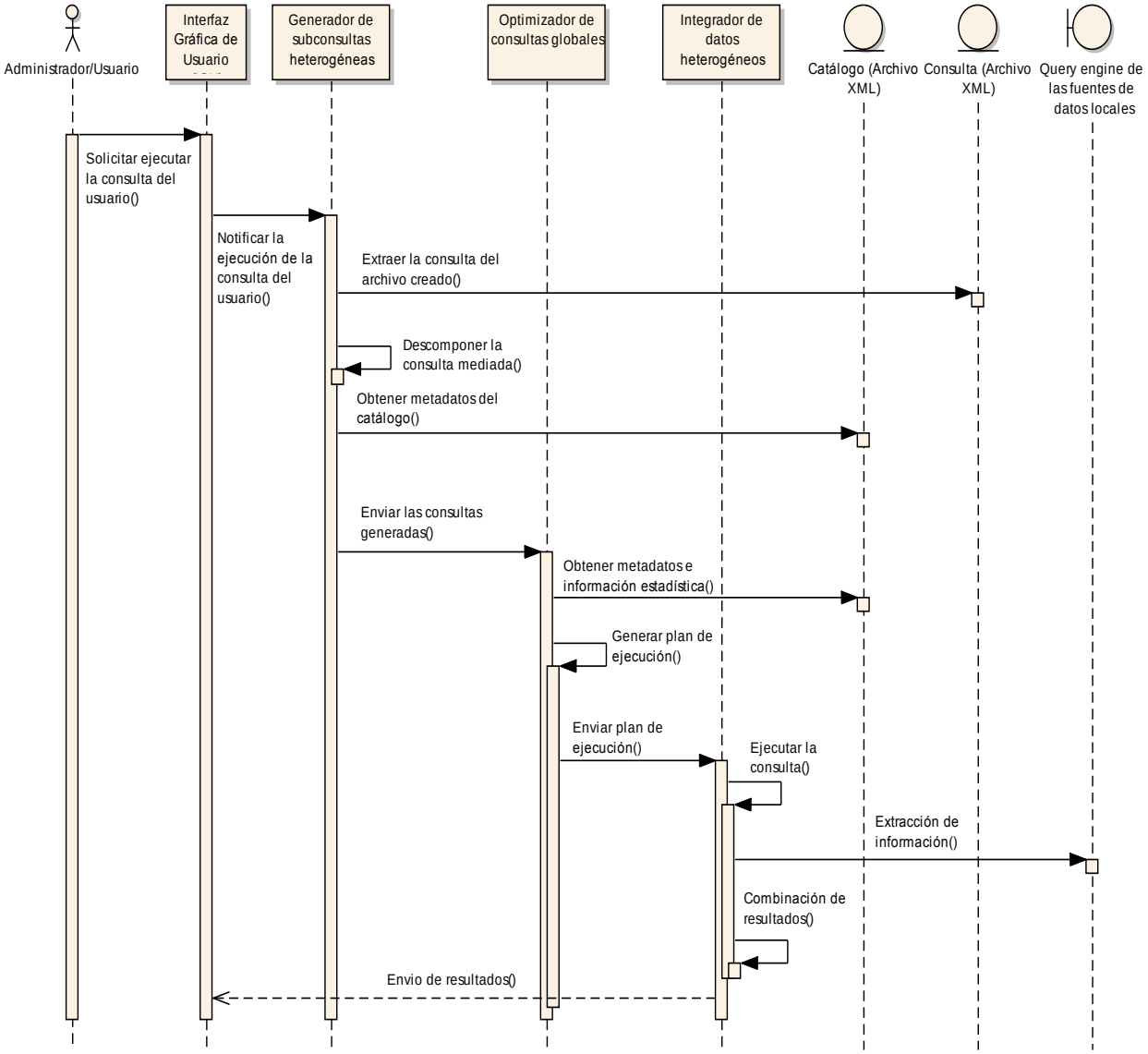


Figura 4.12 Diagrama de secuencia: Ejecución de la consulta mediada.

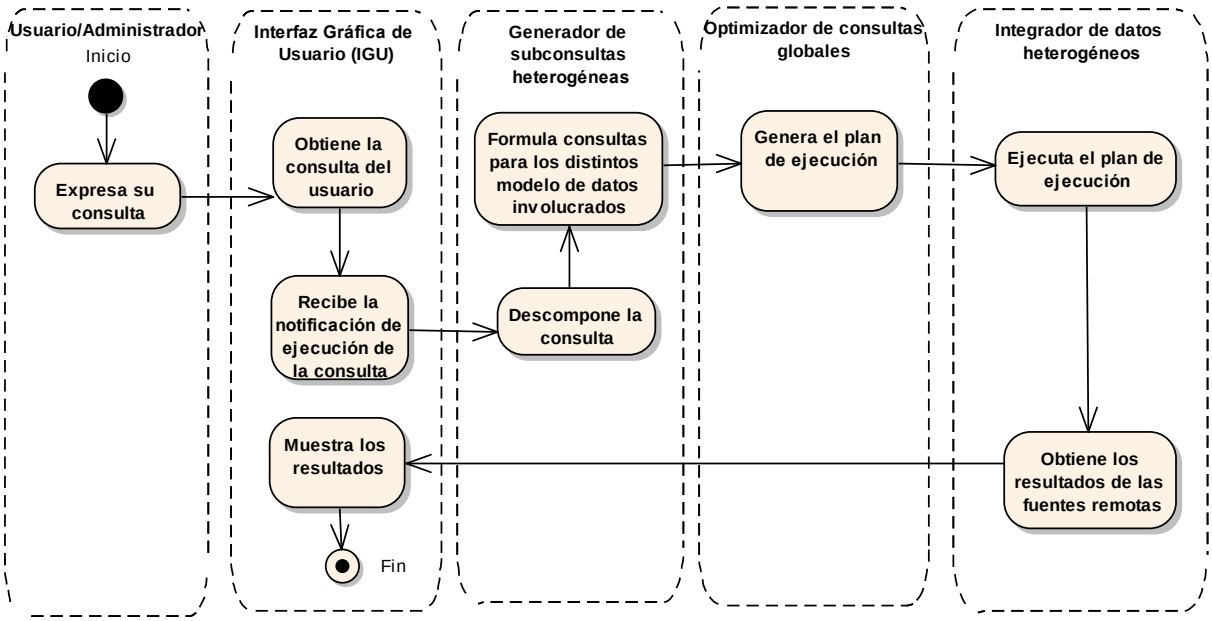


Figura 4.13 Diagrama de actividades: Ejecución de la consulta mediada.

4.4.6. Generación del plan de ejecución.

La función principal del módulo optimizador de consultas globales en un sistema mediador, consiste en la búsqueda del plan de ejecución óptimo, buscando que en su ejecución se beneficie al rendimiento del sistema durante el proceso de integración.

La generación del plan de ejecución inicia en el momento que se recibe la consulta, se utiliza la información contenida en los catálogos de las fuentes de datos buscando correlacionar los atributos mediados con su ubicación física, para ello se crean atributos denominados “Fuente”, que sirven para reconocer las fuentes que podrían resultar útiles paras solucionar la consulta.

El siguiente paso consiste en realizar el “empuje” (desplazamiento) de las operaciones de selección y de condición de la consulta mediada a los atributos “Fuente”; con esto se crean expresiones (consultas objetivo) que están en términos de los esquemas de las fuentes de datos remotas. Una vez que se han completado, se revisa que no existan atributos “Fuente” repetidos; si esta situación ocurre, entonces se fusionan los atributos “Fuente” que consideran como objetivo a la misma base de datos.

Posteriormente se hace uso de las fórmulas estadísticas para calcular el número de elementos que se obtendrán al aplicar las condiciones de filtrado en los accesos a las fuentes de datos remotas, y se realizan estimaciones sobre el número de tuplas que resultaran al hacer las combinaciones necesarias para obtener respuesta a la consulta.

Finalmente, mediante la incorporación de un algoritmo *Greedy* cuya función de progreso pondera a las operaciones de reunión por el número de tuplas resultantes en su realización (auxiliándose para ello de las estimaciones anteriormente mencionadas), se genera un orden, que se incorpora al plan de ejecución y se envía al MIDH para su realización.

A continuación se muestra el proceso anteriormente descrito de una manera sintetizada utilizando un caso de uso y los diagramas de secuencia y de actividades (Véase Tabla 4.6, Figura 4.14, Figura 4.15).

Tabla 4.6 Caso de uso: Generación del plan de ejecución.

Actores participantes	**Interfaz Gráfica de Usuario:** Proporciona la consulta del usuario al sistema mediador. **Módulo de integración de datos heterogéneos (MIDH):** Recibe el plan de ejecución generado.
Condiciones iniciales	• Conjunto de correspondencias. • La definición del esquema mediado. • La consulta del usuario.
Flujo de eventos	1. El módulo OCG recibe la consulta y crea una copia (utilizada como la estructura base para desarrollar el plan de ejecución). 2. Por cada atributo de la consulta mediada, se agrega la información de las ubicaciones físicas con las que este tiene relación. 3. La información referente a la ubicaciones físicas es complementada con sus respectivos datos de conexión (se encuentran contenidos en los catálogos). 4. Al plan de ejecución se le agregan atributos denominados “Fuente”, que se basan en la fuente de datos física a la que pertenecen los atributos de la consulta del usuario, para conocer las fuentes de datos que podrían ser útiles para obtener la información requerida, y posteriormente se utilizan para agrupar las consultas para cada fuente de datos (consultas objetivo).

	5. De la consulta mediada, se transfieren todos los atributos seleccionados a su respectivo atributo “Fuente”, con el fin de generar consultas en términos de las fuentes remotas. 6. De la consulta mediada, se transfieren las condiciones de filtrado a su respectivo atributo “Fuente”, con el fin de generar consultas en términos de las fuentes remotas. 7. Se analiza que no haya elementos “Fuente” repetidos, en caso de haberlos se creando uno nuevo; al terminar los atributos “fuente” que permanecen son considerados como relevantes para resolver la consulta. 8. Se realizan los cálculos estadísticos por cada nodo fuente (consulta local), con el fin de estimar el número de tuplas que resultará por cada consulta local. 9. Se realiza una estimación de resultados intermedios de las posibles combinaciones de las reuniones que se tienen que realizar. 10. Se selecciona el orden que da prioridad a la ejecución temprana de las operaciones que generan una menor cantidad de tuplas. 11. Se envía el plan de ejecución generado al MIDH, para que este lo ejecute.
Condiciones de salida	Se crea el plan de ejecución (en un archivo XML, en los apéndices se anexa un ejemplo de su estructura), en el que se plantea la forma y el orden en el que se deben ejecutar los accesos a las fuentes de datos, buscando proporcionar un rendimiento eficiente al proceso de integración.

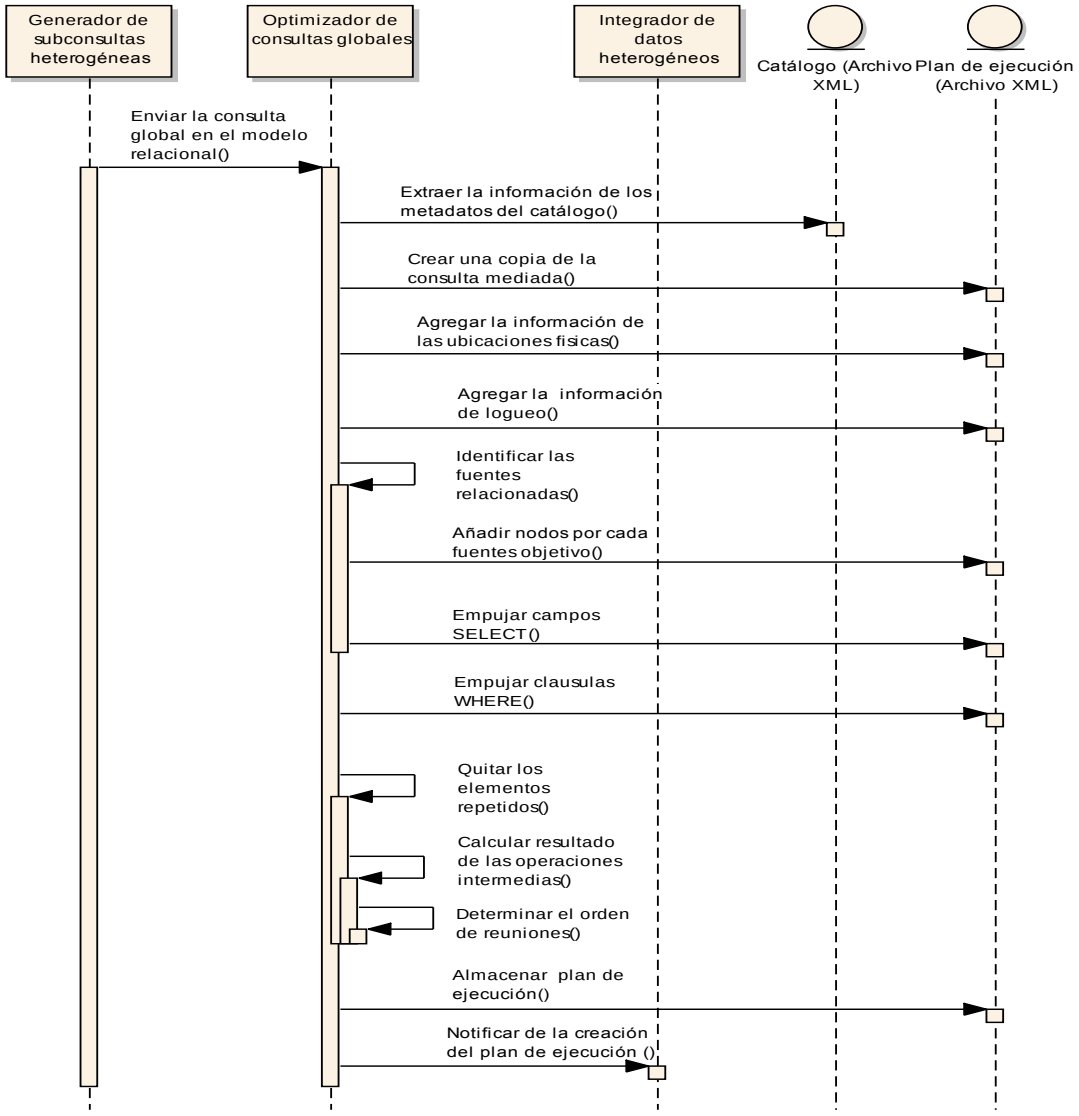


Figura 4.14 Diagrama de secuencia: Generación del plan de ejecución.

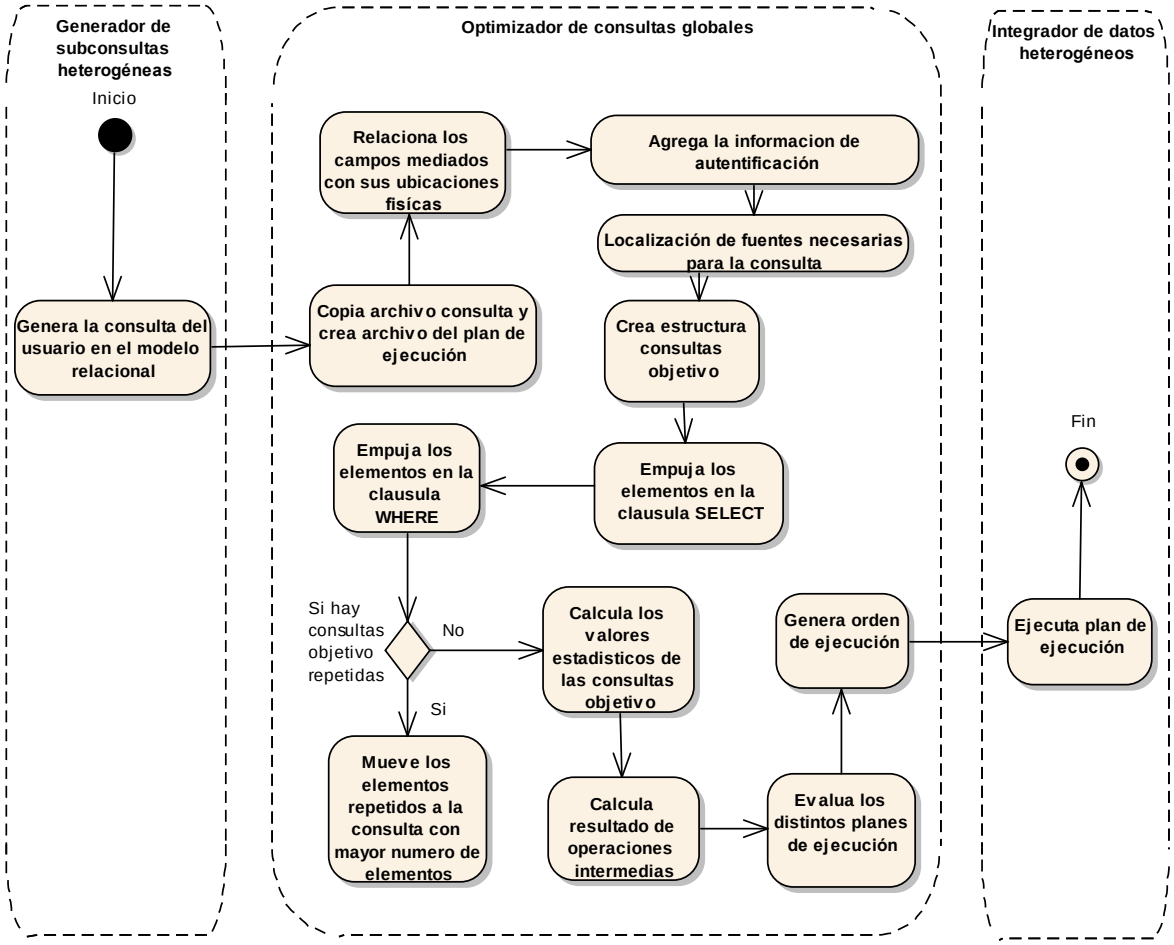


Figura 4.15 Diagrama de actividades: Generación del plan de ejecución.

4.5. Diagrama de clases.

A continuación, se muestra el diagrama de clases (Véase Figura 4.16) de la implementación de la propuesta, en donde se puede ver que maneja una clase como elemento base (llamada "principal") y que esta se auxilia de otras clases ("Crear conexión", "Generar catalogo", "Algoritmo matching", "Crear consulta" y "Generar plan de ejecución") que están especializadas para realizar las tareas anteriormente descritas.

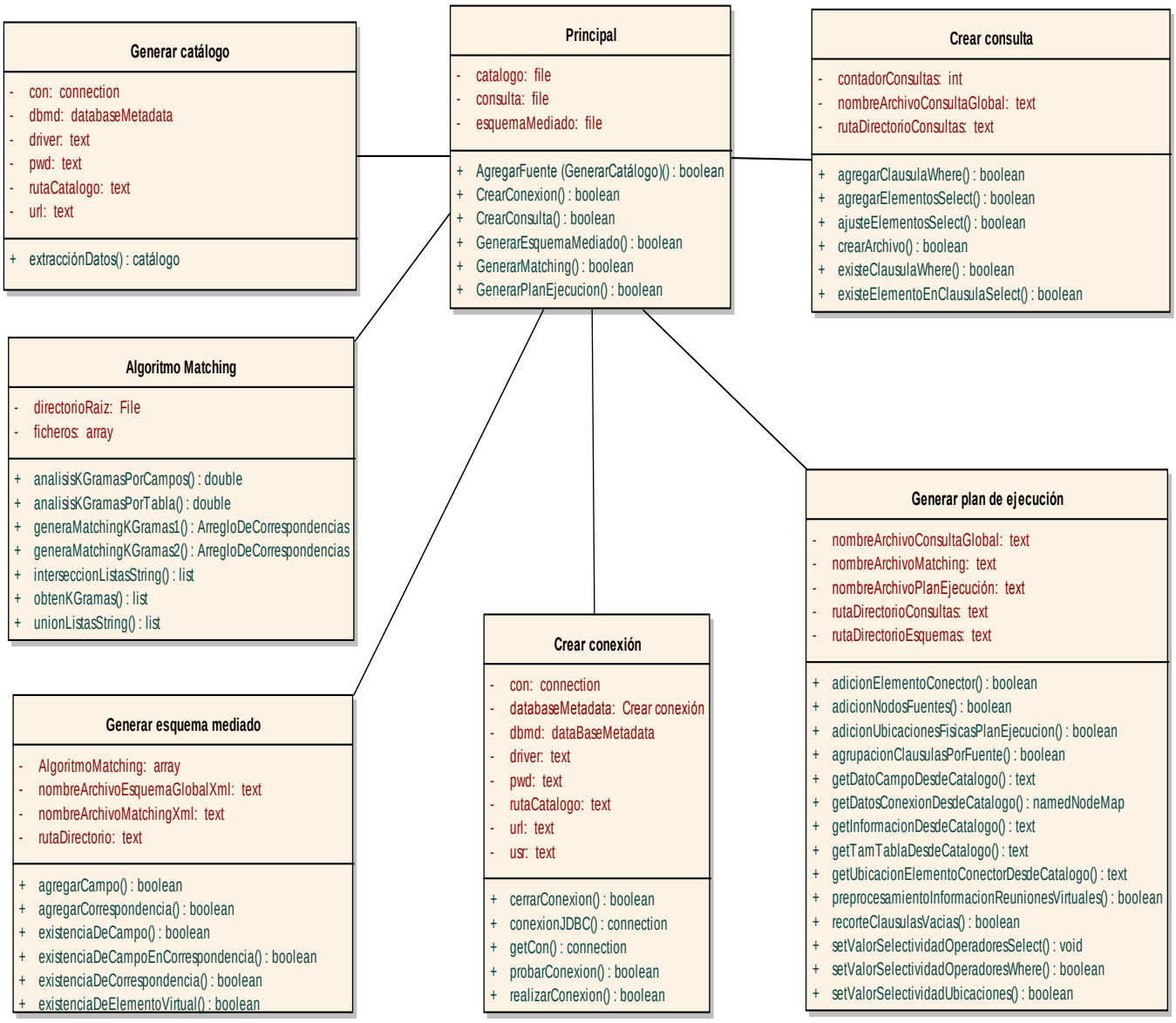


Figura 4.16 Diagrama de clases: Clases utilizadas en el prototipo desarrollado.

Capítulo 5 Implementación.

5.1. Introducción.

Los sistemas mediadores incorporan un módulo optimizador de consultas globales, buscando con ello que el procesamiento de las consultas se efectúe eficientemente, es decir, estas deberán ser resueltas en un tiempo aceptable; sin embargo, existen varios factores que complican este propósito:

- El mediador no materializa la información, esta reside en las fuentes de datos operativas, por ello en tiempo de consulta se realiza tanto la extracción y combinación de los datos, lo que significa tiempo de procesamiento de estas actividades repercutirá en el tiempo de espera del usuario para obtener la respuesta a su consulta.

- Al llevar a cabo la extracción de datos usando la red de comunicaciones como medio de transmisión, el factor de la red afecta de gran manera el tiempo total de respuesta del sistema; no obstante, el estado de la red no es controlable y puede resultar complicado e incluso impráctico tratar de calcularlo.

- Dada la autonomía de las fuentes de datos remotas, el sistema mediador es el único elemento que es capaz de procesar los datos provenientes de dichas fuentes.

A pesar de lo anteriormente expuesto, en este trabajo se desarrolla un software destinado a la generación de planes de ejecución que será incorporada en el módulo optimizador de consultas globales de un sistema mediador, buscando resolver dos elementos importantes en la creación de dichos planes: la descomposición de la consulta mediada en consultas objetivo, que están destinadas a ser ejecutadas en las fuentes de datos remotas, y definir el orden en que se combinarán los datos obtenidos de las consultas, dando prioridad a ejecutar inicialmente las operaciones que produzcan una menor cantidad de tuplas, buscando reducir el procesamiento realizado por el mediador y por consiguiente, el tiempo de espera del usuario para recibir la respuesta a su consulta.

La tecnología usada para la implementación fue Java, debido a que esta ofrece múltiples ventajas, como lo es la robustez, la disponibilidad de un amplio conjunto de bibliotecas, que permiten añadir casi cualquier tipo de funcionalidad, y la razón más importante fue que las aplicaciones desarrolladas con esta tecnología son indiferentes a la arquitectura, dado que Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix, Windows, Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.

A continuación, se describirá la metodología utilizada para resolver esta problemática y cómo se ha incorporado al prototipo desarrollado.

5.2. Condiciones de entrada y salida del módulo optimizador de consultas globales.

El funcionamiento del prototipo está dividido en dos etapas: antes y en tiempo de consulta. La primera etapa hace mención a los elementos que deben ser definidos o proporcionados antes de poder llevar a cabo la generación de los planes de ejecución, mientras que la etapa en tiempo de consulta consiste de los elementos involucrados una vez que el usuario ha expresado su consulta; a continuación se muestran los elementos involucradas en estas etapas (Véase Figura 5.1).

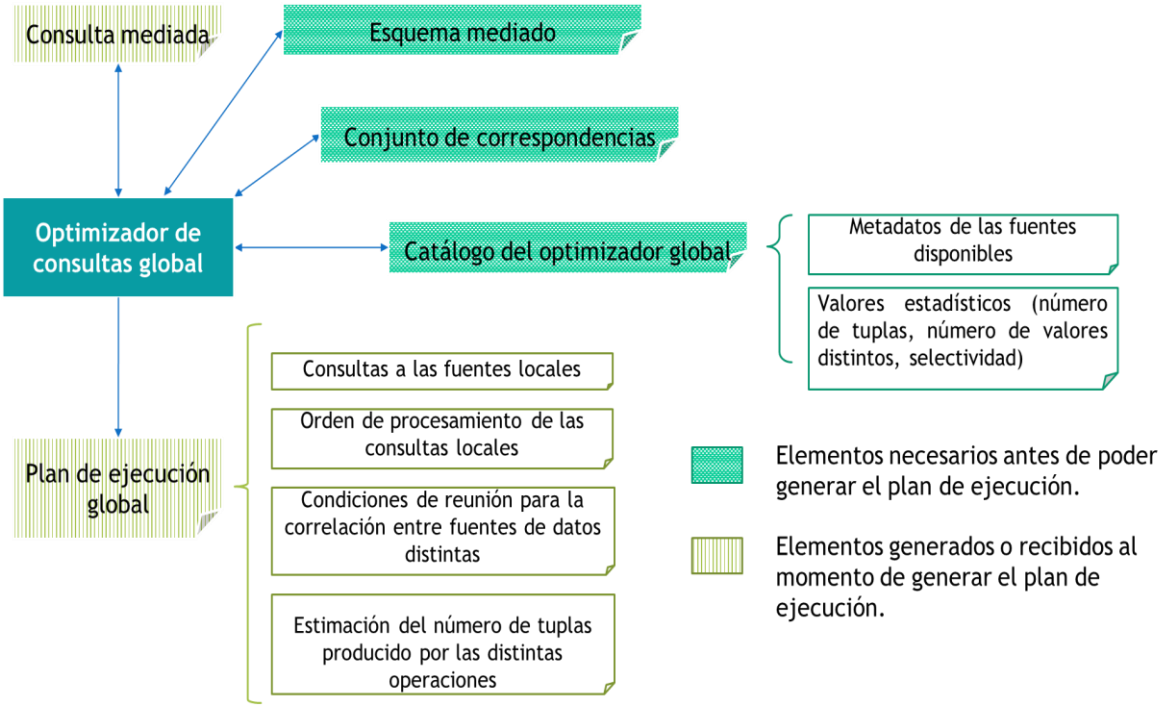


Figura 5.1 Diagrama entradas y salidas del módulo optimizador de consultas globales.

5.2.1. Condiciones de entrada.

A continuación se muestran los elementos necesarios antes de poder recibir la consulta.

- **Los catálogos de información:** La información que describe la estructura y el contenido de las fuentes de datos que se desean integrar.

- **El conjunto de correspondencias:** Las correspondencias son los elementos que indican los atributos que son afines entre las distintas fuentes de datos.

- **La definición del esquema mediado:** contiene los mapeos definidos por el administrador, los cuales sirven para relacionar el esquema mediado con las fuentes de datos locales.

En tiempo de consulta es necesario proporcionar:

- **La consulta del usuario:** incluye los atributos que el usuario desea visualizar y las condiciones de filtrado que deben cumplir los registros para ser considerados relevantes para satisfacer su necesidad de información

Es importante señalar que los elementos anteriormente mencionados deben encontrarse en una representación de árbol en un archivo XML, en donde la estructura de cada uno de estos elementos se expondrá a detalle más adelante.

5.2.2. Condiciones de salida.

Mediante la ejecución de la solución desarrollada se genera como resultado un plan de ejecución global, que incluye:

- **Las consultas objetivo:** Son expresiones que están formuladas en términos de las fuentes de datos remotas, que son consideradas relevantes para cubrir la consulta del usuario y son ejecutadas de manera local por estas.

- **Condiciones de reunión:** Son aspectos necesarios para llevar a cabo la combinación de los datos, con el fin de correlacionar la información de las distintas fuentes mediante un elemento en común y con ello generar información y no solo realizar una combinación aleatoria de datos.

- **Orden de ejecución:** Secuencia lógica en la que deben de ejecutarse las consultas objetivo, buscando que su realización beneficie al rendimiento del sistema mediador durante el proceso de integración.

5.3. Funcionalidades provistas al prototipo desarrollado.

Pese a que el presente trabajo está enfocado al desarrollo de una herramienta de software para la generación de planes de ejecución, proceso que suele operar sin intervención en un sistema mediador, el prototipo también incorpora una interfaz gráfica (Véase Figura 5.2 y Tabla 5.1) que ofrece las siguientes funcionalidades:

- Extracción de los catálogos de las fuentes de datos.
- Generación de correspondencias entre las fuentes de datos.
- Definición del esquema mediado.
- Captura de la consulta del usuario.

- Generación del plan de ejecución óptimo.

Más adelante se mostrará en qué consisten estas funcionalidades:

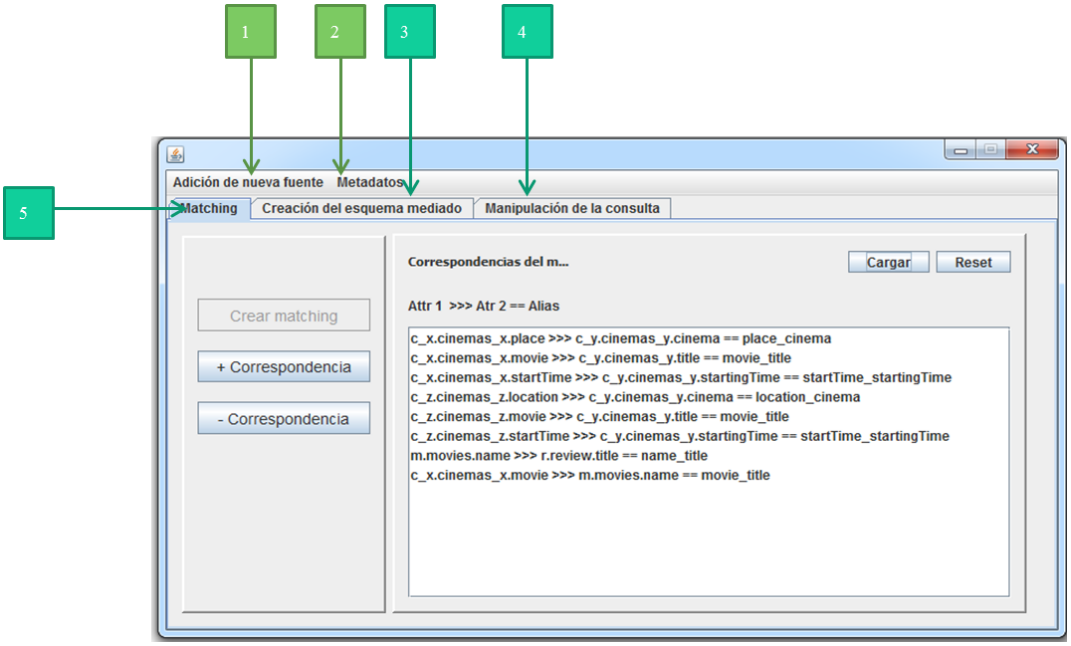


Figura 5.2 Ventana principal del prototipo desarrollado.

Tabla 5.1 Elementos de la ventana principal del prototipo desarrollado.

Elemento	Descripción
1	**Menú "Adición de nueva fuente".** ❖ **"Creación de catálogos":** Este botón invoca la aparición de una ventana, que le permite al administrador probar el estado de conexión con las fuentes de datos remotas y posteriormente llevar a cabo la creación del catálogo de dichas fuentes.
2	**Menú "Metadatos".** ❖ **"Análisis de metadatos":** Al presionar este botón se abre una ventana, donde se exhibe la información contenida en los catálogos y esta le permite al administrador realizar pruebas del algoritmo de análisis de similitud, de tal manera que este puede especificar a qué nivel realizará el análisis de similitud (por tablas campos).

3	**Pestaña esquema mediado:** En este apartado se le permite al administrador definir el esquema mediado, auxiliándose de los catálogos de las fuentes de datos remotas.
4	**Pestaña consulta:** Esta sección está destinada a exhibir los elementos del esquema mediado y proporcionar al usuario de un medio para formular sus consultas.
5	**Pestaña *matching*:** En esta pestaña, se permite al administrador ejecutar el algoritmo de análisis de similitud y de complementarlo mediante la adición y eliminación manual de correspondencias por parte del administrador**.**

5.4. Creación de los catálogos de las fuentes de datos.

El catálogo se utiliza para almacenar información referente a los esquemas y elementos que describen el dominio de los atributos. Este elemento es de gran importancia para el desarrollo de planes de ejecución en un sistema mediador, dado que la información contenida por el catálogo es el elemento del que dispone el mediador para conocer las características de las fuentes remotas a las que tiene acceso.

Para acceder al apartado dedicado a la creación de los catálogos en el prototipo, simplemente es necesario dar clic en el botón “Adición de nuevas fuentes” de la barra de menús y después dar clic en la opción “Creación de catálogo”, esto invocará una nueva ventana (Véase Figura 5.3).

Posteriormente, para que el prototipo pueda acceder a una nueva fuente de datos y crear su respectivo catálogo, es necesario que el administrador proporcione la URL de la fuente de datos, el nombre de usuario, la contraseña, el controlador y dar clic en el botón “Creación de catálogo”

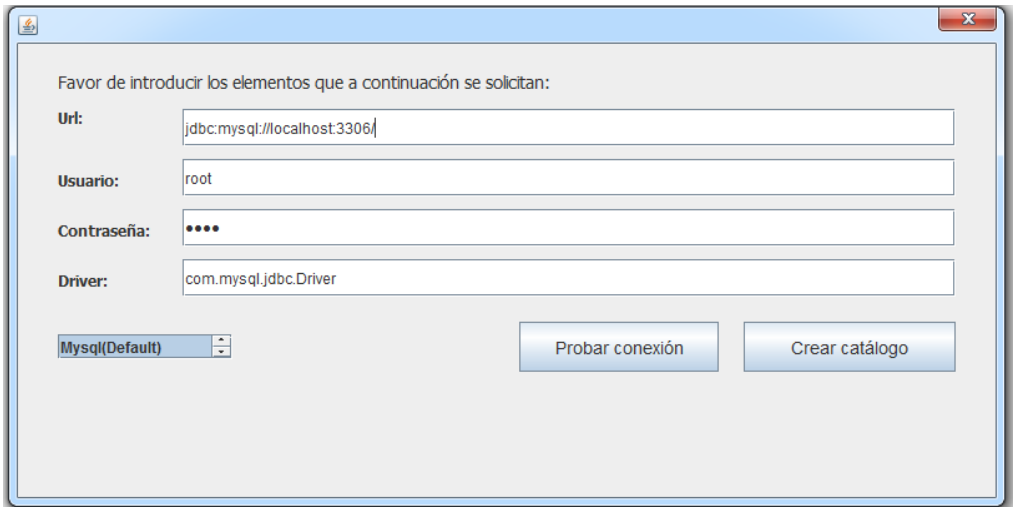


Figura 5.3 Adición de una nueva fuente de datos y creación de su catálogo.

Una vez que se da la instrucción de llevar a cabo la creación del catálogo de una fuente de datos en específico, el prototipo utiliza JDBC para establecer una conexión con el SABD (Sistema administrador de bases de datos) de la fuente de datos remota, esto desencadena la ejecución de la siguiente rutina:

> Inicialmente los datos que se usaron para establecer la conexión se recuperan y almacenan provisionalmente, después se extraen los metadatos de los esquemas utilizando una interface provista por el JDBC, finalmente para obtener algunos metadatos y valores estadísticos que no son disponibles mediante dicha interface, como lo son el número de valores distintos, el valor de selectividad, los valores máximo y mínimo, entonces se ejecutan consultas de agregación que consideran a los esquema contenidos y al propio SABD de la fuente de datos remota, auxiliándose de los metadatos obtenidos y de la conexión establecida, por ejemplo: para conocer el número de valores distintos, se construye y ejecuta una consulta de agregación, que solicita el conteo de los elementos únicos de determinado campo, este proceso se repite por cada atributo, de cada tabla y de cada base de datos. Después de haber obtenido la información de la fuente de datos remota, se almacena en un archivo, se optó por utilizar XML para el formato de almacenamiento.

La información almacenada en el catálogo puede clasificarse en tres categorías:

- **Información de autentificación:** Son los datos necesarios para establecer una conexión con alguna de las fuentes de datos remotas; es necesario que por cada fuente de datos distinta se disponga del nombre de la base de datos, el nombre de usuario, la contraseña, el controlador de conexión y la URL.

- **Metadatos:** Es la información que representa los esquemas de las fuentes remotas (bases de datos, las tablas y atributos), además de la información descriptiva importante como la siguiente:
  - *Por cada tabla:* el número de registros, las llaves primarias y las llaves foráneas.
  - *Por cada atributo:* la clave del tipo de dato, el nombre del tipo de dato, el tipo de dato SQL, el valor máximo, el valor mínimo y si el atributo permite nulos.

- **Valores estadísticos:** Se le denomina de esta manera a los elementos que describen la distribución de los valores contenidos en cada campo, como son el número de valores distintos y la selectividad.

5.5. Generación de correspondencias entre las fuentes de datos.

Previamente se ha mencionado que el catálogo es el elemento que provee de la información de las fuentes remotas al sistema mediador; sin embargo, este no incluye información alguna sobre la relación que mantienen los datos contenidos entre las distintas fuentes de datos, por ello se tiene que hallar y definir dicha relación, para que posteriormente se pueda establecer el esquema mediado.

La correlación de esquemas distintos o *schema matching*, resulta ser un problema complicado que requiere de todo un proceso de análisis (que no está contemplado en los alcances de este trabajo), en el caso más simple y bajo un pensamiento optimista se puede suponer que el nombre de los campos que comparten un mismo dominio serán subcadenas uno de otro, por ejemplo: para almacenar el nombre de un producto podría pensarse que se utilizaría “nombre_producto”, “nombreProducto”, “nom_Prod”, “nomProd”; pero podría ocurrir que los nombres de los atributos afines estén almacenados usando sinónimos o abreviaciones. Este tipo de circunstancias dificulta la labor del administrador para localizar las correspondencias existentes entre los esquemas, por ello con el fin de facilitar esta labor al administrador, el prototipo desarrollado incorpora un algoritmo de matching que realiza un análisis de similitud sobre los metadatos (nombres de los campos) utilizando el coeficiente de Jaccard mediante el traslape de k-gramas (información referente a estos elementos será provista en las siguientes páginas).

Este algoritmo tiene dos debilidades: no considera el contenido de las fuentes de datos ni tampoco realiza un análisis de los sinónimos o abreviaturas de los metadatos, por ello el conjunto de correspondencias generadas por este algoritmo requiere ser validado y complementado por el administrador.

A continuación se muestra la ventana asociada a la generación de correspondencias (Véase Figura 5.4), en esta se permite ejecutar el algoritmo de análisis de similitud (botón “Crear matching”), así como añadir y eliminar elementos de manera manual (botones “+Correspondencia” y “-Correspondencia”).

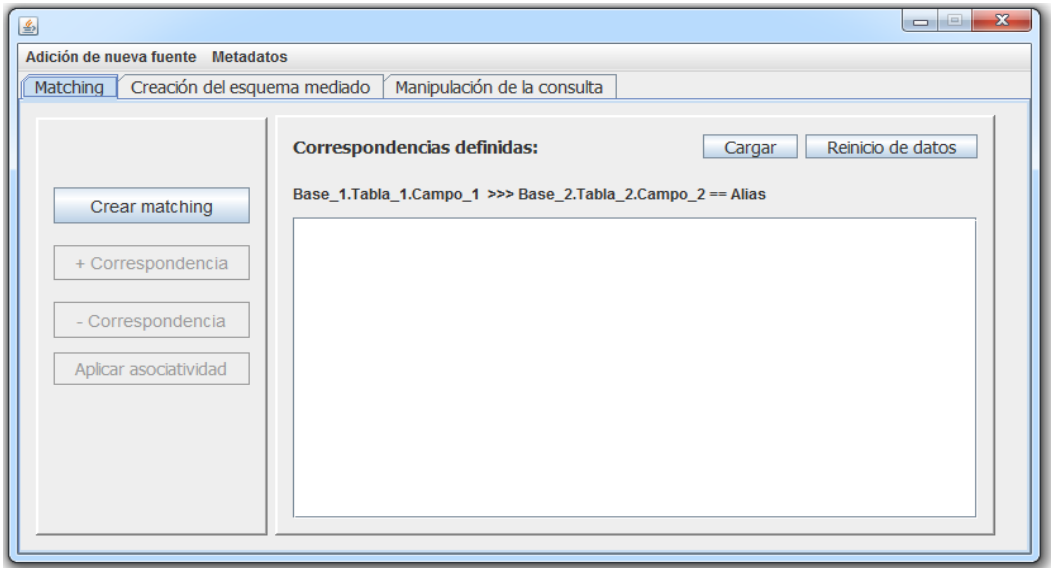


Figura 5.4 Ventana principal controles usados para el manejo de correspondencias.

La aplicación desarrollada también incorpora una ventana (Véase Figura 5.5) que permite ejecutar el algoritmo de correspondencias (*matching*) sobre tablas y atributos en específico; es accesible al dar clic en el botón "Metadatos" del área de menús de la ventana principal y después al dar clic en el botón "Análisis de metadatos".

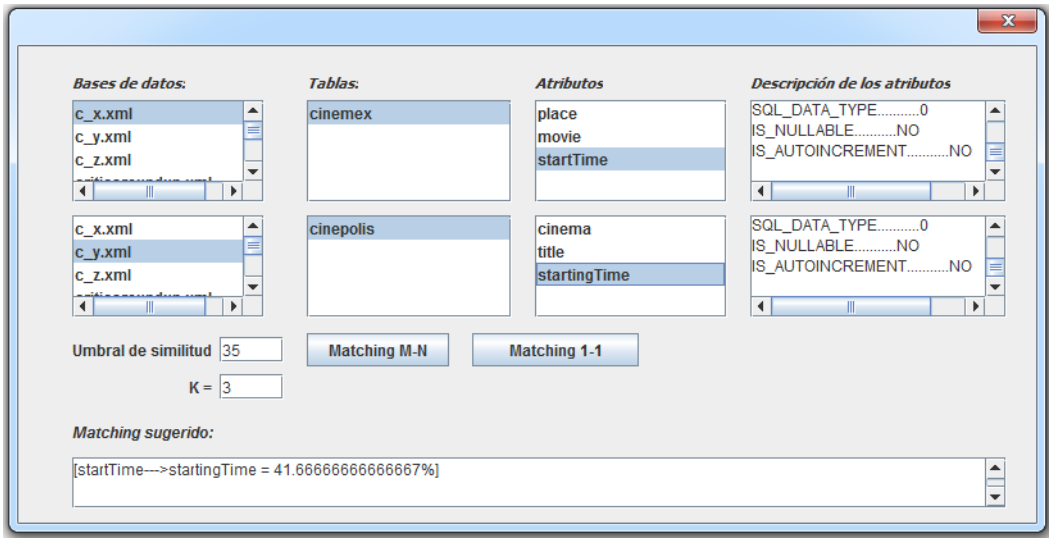


Figura 5.5 Análisis de metadatos y del algoritmo de similitud.

5.5.1. K-gramas.

El análisis de k-gramas suele ser utilizado por sistemas de recuperación de información para encontrar palabras que podrían necesitar de una corrección ortográfica, sistemas de reconocimiento de patrones que buscan estimar la probabilidad de que una palabra aparezca en

un texto y en sistemas de integración ha demostrado ser útil como *matcher (quien genera las correspondencias)* para localizar las correspondencias existentes entre esquemas distintos.

El termino k-grama, hace referencia a la fragmentación de una palabra en partes de dimensión igual a k, mediante un proceso de barrido iterativo por cada carácter hasta formarse el ultimo k-grama.

A continuación (Véase Tabla 5.2), se muestran los k-gramas de la palabra "manzana" para k =2, k = 3 y K = 4:

Tabla 5.2 Ejemplo de uso de k-gramas.

Valor de K	Fragmentos de la palabra (k-gramas)
2	Ma/an/nz/za/an/na
3	Man/anz/nza/zan/ ana
4	Manz/anza/nzan/zana

5.5.2. Índice de Jaccard.

El índice de Jaccard o también conocido como coeficiente de similitud (nombrado originalmente como *coefficient de communauté* por Paul Jaccard) es un mecanismo estadístico que suele ser utilizado para realizar el análisis de similitud y de la diversidad en conjuntos de prueba finitos.

Está definido como la división de la intersección sobre la unión de los elementos de los conjuntos analizados (Véase Figura 5.6).

$$J(A,B) = \frac{[A \cap B]}{[A \cup B]}$$

Figura 5.6 Fórmula del índice de Jaccard.

Al manejarse palabras como elemento de análisis, es necesario dividirlas en fragmentos para poder obtener su representación como conjuntos, para ello puede utilizarse la descomposición en k-gramas. Una vez que se tienen ambos conjuntos, se realiza la unión para generar uno nuevo (sin añadir elementos repetidos), y después se realiza la intersección de ambos conjuntos en un conjunto nuevo (recordando que esta es definida como los elementos que son contenidos por ambos conjuntos), y finalmente para obtener el valor numérico del coeficiente de similitud se realiza la división del grado de la intersección entre el grado de la unión.

Por ejemplo si se desea hacer la comparación de las palabras "Ubicación" y "Dirección" utilizando k = 3, se tiene (Véase Tabla 5.3):

Tabla 5.3 Ejemplo del coeficiente de Jaccard.

Palabras	Descomposición en k-gramas (k=3)	Unión	Intersección	Coeficiente de Jaccard
Ubicación	Ubi bic ica cac aci **cio ion**	ubi bic ica cac aci dir ire rec ecc cci **cio ion** (12)	**cio ion** (2)	2/12 = 0.166 16.6%
Dirección	Dir ire rec ecc cci **cio ion**			

5.5.3. Análisis de similitud de metadatos usando el índice de Jaccard.

El algoritmo de análisis de similitud de los metadatos usando el índice de Jaccard, tiene como objetivo proporcionar un conjunto de correspondencias de los elementos que manejan un mismo dominio, pero debido a que este matching se realiza considerando únicamente la similitud entre los nombres de los campos, requiere ser validado por el administrador.

Para poder ejecutarse, es necesario contar con un valor de umbral (definido por el administrador) para utilizarlo como criterio de evaluación de las correspondencias sugeridas, y manejar la información de los esquemas de las fuentes remotas.

Posteriormente, mediante la utilización de una serie de bucles anidados, se calcula el valor del índice de Jaccard tomando en cuenta los nombres de los atributos de cada tabla y de cada fuente con respecto a los nombres de los atributos de cada tabla de las demás fuentes de datos; si el valor obtenido es mayor o igual al umbral definido por el administrador, entonces se almacena la información referente a dicha correspondencia, en caso contrario se descarta.

Buscando dar mayor claridad a la importancia de la selección del valor de umbral, suponga que un conjunto de fuentes de datos autónomas existe un atributo que hace mención al “nombre de un producto” y este es representado con las siguientes variantes de nombre en las fuentes de datos:

- Nombre
- Producto
- NombreProducto
- NombreDelProducto
- Nombre_Producto
- Nombre_Pdto
- Nom_Pdto

Al realizar un análisis de los niveles de similitud obtenidos al evaluar las alternativas posibles se obtienen los siguientes resultados (Véase Tabla 5.4) (Para un mayor detalle en la utilización del análisis de similitud de esquemas véase el Apéndice F):

Tabla 5.4 Ejemplo del análisis de similitud para las variantes del campo "nombre de producto".

Id	$Campo_1$	$Campo_2$	Similitud obtenida utilizando el índice de Jaccard (k = 3)
1	Nombre	Producto	0%
2	NombreDelProducto	Nom_Pdto	5%
3	Nombre_Producto	Nom_Pdto	5.55%
4	Nombre_Pdto	Nom_producto	5.55%
5	NombreProducto	Nom_Pdto	5.88%
6	Nombre	Nom_Producto	7.69%
7	Nombre	Nom_Pdto	11.11%
8	NombreDelProducto	Nombre_Pdto	20%
9	Nom_Pdto	Nom_producto	23.07%
10	NombreProducto	Nombre_Pdto	23.52%
11	Nombre	NombreDelProducto	26.66%
12	Nombre	Nombre_Producto	30.76%
13	Nombre	NombreProducto	33.33%
14	Nombre_Pdto	Nom_Pdto	36.60%
15	Nombre_Producto	Nombre_Pdto	37.50%
16	NombreDelProducto	Nom_Producto	38%
17	Nombre	Nombre_Pdto	44.44%
18	NombreProducto	Nom_producto	46.66%
19	Nombre_Producto	Nom_producto	53.33%
20	NombreDelProducto	Nombre_Producto	55.55%
21	NombreProducto	NombreDelProducto	58.82%
22	NombreProducto	Nombre_Producto	66.66%

Puede afirmarse que es de gran importancia definir un valor de umbral apropiado, debido a que este define el nivel de flexibilidad que se manejará para poder considerar dos campos como afines.

- Si se exige un nivel de umbral demasiado alto, puede suponerse que el usuario visualizará un número reducido de las posibles correspondencias, dado que el algoritmo únicamente consideraría como correspondencias aquellas con elementos prácticamente idénticos.

- Por el contrario, si el valor de umbral demasiado bajo, entonces puede suponerse que el usuario visualizará un gran número de posibles correspondencias, que probablemente no tengan relación alguna.

El proceso anterior se describe con el siguiente algoritmo (Véase Figura 5.7):

Algoritmo: Análisis de similitud de metadatos usando el índice de Jaccard.

Entrada: Conjunto de catálogos de las bases de datos accesibles (C).
Salida: Conjunto de correspondencias sugeridas como matching (M).

```
1   U =  Valor umbral de similitud  definido por el administrador.
2   M  ←0                          // Archivo que almacena al conjunto de correspondencias.
3   s =0                           //Resultado del análisis de similitud.
4   kgramas_1= {},  kgramas_2= {}  // Arreglos que almacenan los k-gramas de un campo.
5   for each catálogo_i in C do
6           for each tabla_j in catálogo_i do
7                   for each atributo_k in tabla_j do
8                           for each catálogo_p in C  where catálogo_p != catálogo_i  do
9                                   for each tabla_q in catálogo_p do
10                                          for each atributo_r in tabla_q do
11                                             kgramas_1 = obtencionKgramas(atributo_k)
12                                             kgramas_2 = obtencionKgramas (atributo_r)
13                                             s =  CoeficienteJaccard(kgramas_1, kgramas_2)
14                                         corr = new correspondencia(catálogo_i, tabla_j  atributo_k
                                                                      catálogo_p, tabla_q atributo_r)
15                                             If( (s >= U ) )
16                                                     then  M  ←Corr
17                                             end if
18                                          end for
19                                  end for
20                          end for
21                  end for
22          end for
23  end for
24  return M
```

Figura 5.7 Algoritmo del análisis de similitud de metadatos usando el índice de Jaccard.

5.6. Definición del esquema mediado.

Como se ha mencionado previamente, el esquema mediado es la representación de los elementos que puede visualizar el usuario y sobre los cuales tiene permitido expresar sus consultas.

Para el administrador que tiene la responsabilidad de crear el esquema mediado, este no es únicamente una versión resumida y simplificada de la información proveniente de las fuentes de datos autónomas, sino que también implica un conjunto de mapeos, que determinan la relación de este esquema con los esquemas de las fuentes de datos remotas.

En el esquema mediado se manejan dos conceptos importantes que se deben tener claros.

- **Entidad virtual:** Hace referencia a las relaciones que existen en el esquema mediado y que son el resultado dinámico de una o más operaciones relacionales que operan sobre las relaciones base para generar otra relación.

- **Atributo virtual:** Es semejante a lo que sería un atributo en una base de datos centralizada, solamente que este pertenece al esquema de una entidad virtual.

Por ejemplo: Si se desea crear un esquema mediado y se tiene acceso a los siguientes esquemas de las fuentes de datos locales:

- F.Filmes(nombre, protagonista, director, genero)
- R.Reseñas(titulo, añoEstreno, puntuacion, resumen)
- X.Cine_1(nombreCinema, nombrePelicula, horaInicio)
- Y.Cine_2(nombre_cine, titulo_pelicula, inicio_funcion)
- Z.Cine_3(cinema, largometraje, funcion)

Entonces una forma de definirlo sería la siguiente (Véase Tabla 5.5):

Tabla 5.5 Entidad y mapeos del ejemplo.

Entidad mediada	Esquemas locales
Peliculas(titulo,director,año,genero)	**F.Filmes**(nombre,director,genero), **R.Reseñas**(titulo,añoEstreno)
Actores(titulo,protagonista)	**F.Filmes** (nombre,protagonista)
Funciones(titulo,cine,funcion)	**X.Cine_1**(nombreCinema,nombrePelicula,horaInicio) **Y.Cine_2**(nombre_cine,titulo_pelicula,inicio_funcion) **Z.Cine_3**(cinema, largometraje, funcion)
Descripcion(titulo, valoracion, reseña)	**R.Reseñas**(titulo, puntuacion, resumen)

La representación conceptual de este esquema mediado sería (Véase Figura 5.8):

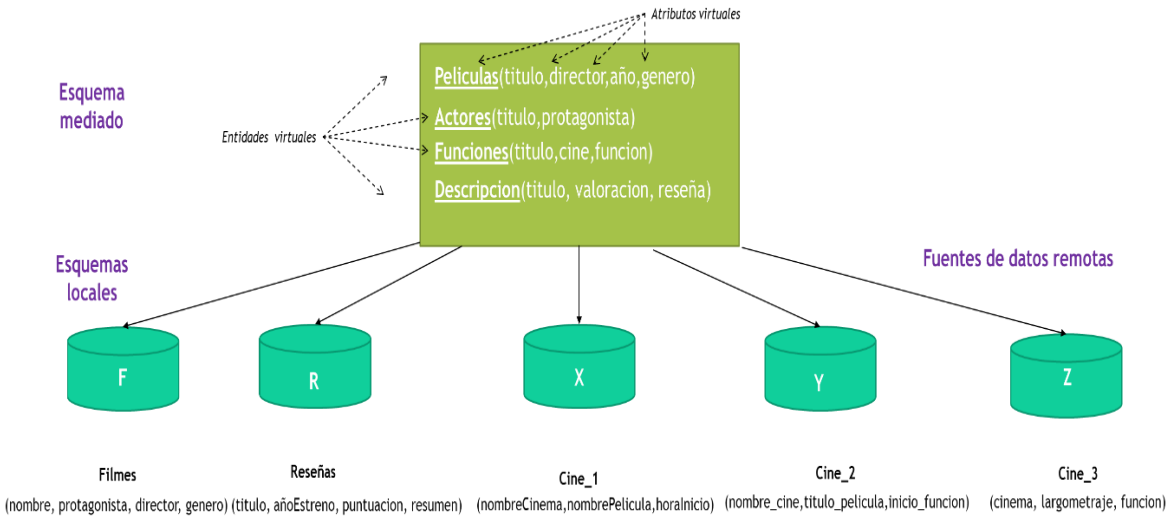


Figura 5.8 Ejemplo de esquema mediado.

El prototipo desarrollado incluye una pestaña (“Creación del esquema mediado”) dedicada a realizar la definición del esquema mediado de una forma rápida y sencilla, permitiéndole al administrador añadir atributos virtuales bajo dos modalidades.

- **Atributo simple:** Es aquel que como su nombre lo indica, se encuentra conformado por la referencia de un atributo; sin embargo debe de existir previamente una correspondencia que haga mención a la fuente de datos a la que pertenece dicho atributo (Véase Figura 5.9).

- **Atributo compuesto** Es aquel cuyo dominio es manejado en múltiples fuentes y por consiguiente es asociado a alguna de las correspondencias previamente definidas (Véase Figura 5.10).

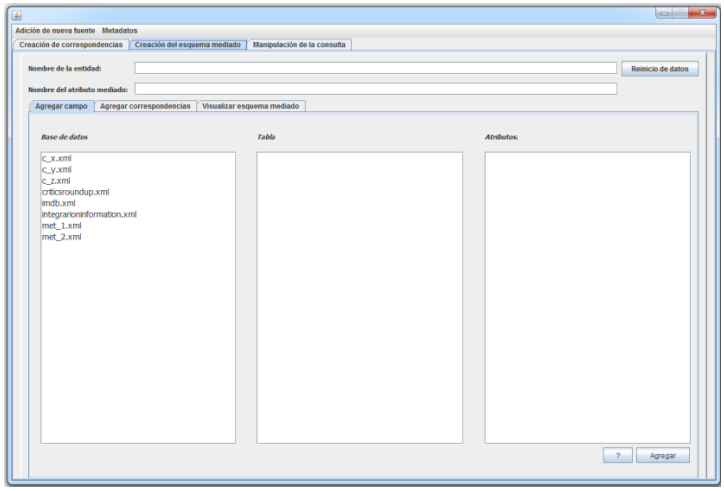


Figura 5.9 Adición de un atributo simple.

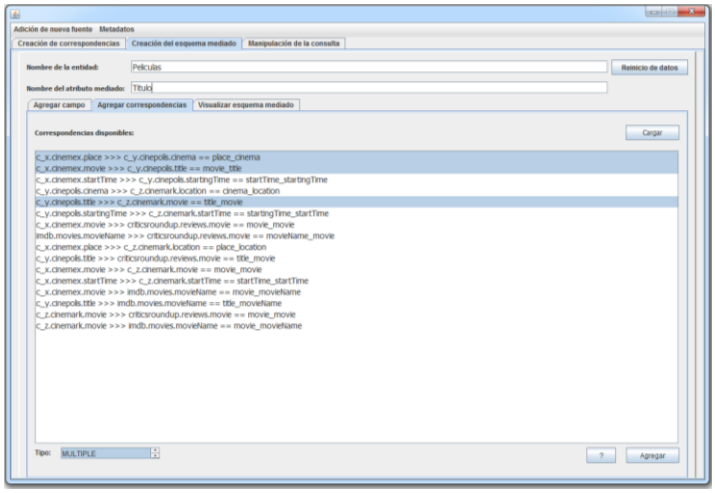


Figura 5.10 Adición de un atributo compuesto.

5.7. Captura de la consulta del usuario.

El prototipo desarrollado provee al usuario de un apartado destinado a la formulación de consultas (Véase Figura 5.11) con el fin de facilitar la interacción del usuario con la aplicación, esta ventana puede invocarse dando clic en el botón "Crear consulta" de la pestaña "Manipulación de la consulta"; en este apartado se visualizan los elementos que componen al esquema mediado, además permite al usuario expresar sus consultas.

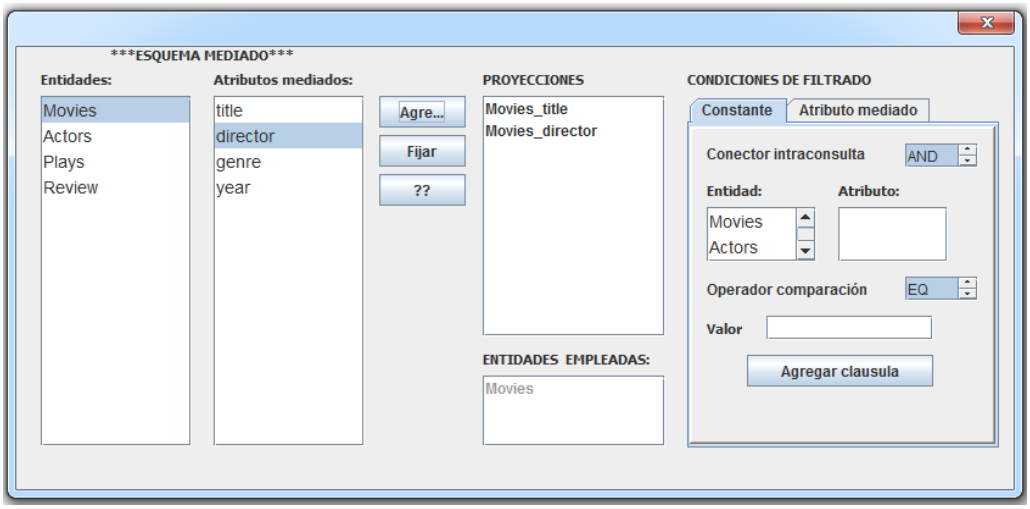


Figura 5.11 Captura de la consulta del usuario.

Las consultas, definidas mediante la interfaz gráfica y por consiguiente que pueden ser procesadas para la generación de su respectivo plan de ejecución, están definidas de la siguiente manera:

- **Proyecciones:** Son los elementos mediados (atributos) que el usuario desea visualizar.

- **Condiciones de filtrado:** Son las condiciones de filtrado (cláusulas) que permiten al usuario la especificación de las características que debe cumplir un registro para satisfacer la consulta del usuario.

  La consulta puede contener múltiples cláusulas conjuntivas, que deben hacer referencia a dos operandos y a un operador, los cuales se especifican de la forma:

  Operando 1 *Operador* *Operando 2*

  De tal forma que:

  - **El operando 1:** hace referencia a un atributo del esquema mediado.
  - **El operando 2:** es un valor constante.

- **El operador:** es el elemento que define el tipo de comparación que se realizará, en donde los elementos aceptados son: igual, distinto, mayor que, mayor o igual que, menor que y menor o igual que.

5.8. Generación un plan de ejecución.

La obtención del plan de ejecución puede ser dividida en dos etapas:

- **Descomposición de la consulta mediada:** El desarrollo de expresiones en términos de los esquemas de las fuentes remotas (consultas objetivo).

- **La asignación de un orden de ejecución:** Buscará que la ejecución de la consulta se realice de manera eficiente.

5.8.1. Descomposición de la consulta mediada.

Los pasos utilizados para la descomposición de la consulta son:

1. Inicialmente es necesario contar una estructura de árbol para expresar el nivel jerárquico del plan de ejecución, por ello se realiza una copia de la consulta del usuario y se procede a utilizarse como elemento base para crear el plan de ejecución, en donde el archivo contiene la consulta de usuario expresada en términos del esquema mediado, incorporando sus respectivos operadores SELECT y WHERE.

2. Se realiza la correlación de cada uno de los atributos virtuales de la consulta mediada con sus ubicaciones físicas (utilizando la información del esquema mediado). Esto se lleva a cabo añadiendo las ubicaciones físicas como información complementaria a los atributos virtuales.

3. Por cada mención hacia una entidad virtual existe un operador que las agrupa, posteriormente se realiza la síntesis de las ubicaciones físicas de los atributos virtuales, mediante la adición de operadores denominados como “Fuente” (estos posteriormente serán necesarios).

4. Se realiza el empuje de cada uno de los elementos de los operadores SELECT y WHERE de la consulta mediada a su respectivo operador “Fuente”, formando expresiones en términos de cada fuente de datos con sus propios operadores SELECT y WHERE.

5. Se realiza el podado de las subconsultas que estaban expresadas en términos de las entidades mediadas y se dejan únicamente los operadores “Fuente”.

6. Se busca que no hayan operadores “Fuente” repetidos, que realicen accesos a una misma fuente de datos, y en caso de existir, se fusionan dejando como resultado una sola expresión.

7. Como resultado de los anteriores pasos, en cada nodo fuente se tiene la información necesaria para realizar consultas a las distintas fuentes de datos implicadas en la consulta del usuario.

A continuación se muestra este proceso de modo resumido mediante su representación como algoritmo (Véase Figura 5.12):

Algoritmo: Descomposición de la consulta mediada.

Entrada: *Consulta mediada (Q_M), Esquema mediado(E)*
Salida: *Conjunto de consultas locales(Q_L)*

```
1   Q_L = Q_M
2       for each consulta de Q_L
3           for each WHERE de consulta_i
4               validaciónJoinMediado (SELECT)
5               validaciónJoinMediado (WHERE)
6           end for
7       end for
8       for each campoMediado de Q_L
9           for each ubicacion_i de campoMediado_i en E
10              campoMediado ← ubicacion
11          end for
12      end for
13      for each consulta de Q_L
14          for each campoMediado de consulta_i
15              for each ubicacion de campoMediado
16                  If(!ubicacion existe en fuente de consulta_i)
17                      fuente ← ubicacion
18                      consulta_i ← fuente
19                  end if
20              end for
21          end for
22      end for
23      for each consulta de Q_L
24          for each campoMediado de consulta_i
25              If(!campoMediado_j existe en SELECT de fuente de consulta_i)
26                  SELECT ← PUSH(campoMediado_j)
27              end if
```

```
28                          If(!campoMediado_j ∈ WHERE de fuente de consulta_i)
29                              WHERE ← PUSH(campoMediado_j)
30                          end if
31                      end for
32                  end for
33  Return Q_L
```

Figura 5.12 Algoritmo de descomposición de la consulta mediada a las consultas locales.

5.8.3. Ordenamiento de las reuniones.

Antes de proceder a ejecutar las consultas obtenidas de etapa anterior (que están en términos de las fuentes de datos locales), debe definirse un orden de ejecución que busque proveer de eficiencia al procesamiento de la consulta.

Los pasos utilizados para asignar el orden de las reuniones son:

1. Esta etapa toma como entrada el plan de ejecución generado por la etapa de descomposición (este únicamente contiene la información para realizar los accesos a las fuentes de datos remotas), no obstante, este no detalla la manera en que se llevaran a cabo, es decir no se especifica el orden ni los atributos que utilizarán en la condición de reunión entre resultados intermedios para hacer la combinación de datos.

2. Se utiliza la información contenida en los catálogos de cada una de las fuentes de datos y de un conjunto de fórmulas para calcular la cantidad de registros que se obtendrán de las fuentes remotas, posteriormente estos valores son utilizados para estimar el número de tuplas resultantes de las distintas alternativas que existen para realizar las operaciones de reunión implicadas en la consulta, y estos valores se almacenan temporalmente.

3. Se realiza la búsqueda de la operación de reunión cuya realización genere la menor cantidad de tuplas; si no se ha definido una operación previa, entonces se define como la primera operación, en cambio si previamente se ha definido alguna operación, entonces se comprueba que la operación analizada involucre a alguna de las fuentes cuyos datos ya se han procesado; si esta condición se cumple, entonces esta operación se considera como la siguiente.

4. Este proceso se repite hasta que se hayan encontrado los elementos de reunión para cada una de las consultas de las que se requiere combinar.

5. Como resultado de este proceso se obtiene el orden de ejecución para la realización de las consultas remotas y se definen los atributos que se utilizaran para la reunión.

A continuación se muestra el algoritmo que realiza este proceso (Véase Figura 5.13):

Algoritmo: Orden de reuniones.

Entrada: *Conjunto de consultas objetivo(Q_L), Matching(M), Lista de fuentes objetivos(L)*
Salida: *Plan de ejecución(P)*

```
1   P ← 0
2   t ← inf
3   c ← 0
4   r ← 0
5   while(!empty(QL))
6   consultai
7           for each consultaj de QL consultaj y que pertenece a L
8                   corr[] = getCorrespondencias(Base de consultai ,Base de consultaj)
9                           for each corr de corr[]
10                                  aux = getTamañoResultado(consultai, consultaj,corrk)
11                                          if(aux < t)
12                                                  c = corr
13                                          end if
14                          end for
15                          L = Remover(Base de consultai)
16                          consultai = r
17                          r++
18                          P ← c
19          end for
20
21  end loop
22  return P
```

Figura 5.13 Algoritmo utilizado para determinar el orden de las reuniones.

Capítulo 6 Pruebas y resultados.

6.1. Introducción.

En este capítulo se presentan las pruebas a las que se sometió el prototipo que incorpora la solución propuesta para la generación de planes de ejecución en un sistema mediador, con el fin de analizar su eficiencia ante distintos tipos consultas en un escenario de integración de información.

6.2. Caso de estudio: "Películas".

El primer caso de estudio analizado plantea un contexto donde se requiere llevar a cabo la integración de información referente a funciones de películas, para proveer de un acceso unificado a la información contenida en fuentes de datos independientes que manejan información relacionada. Buscando liberar al usuario de tener que interactuar con cada una de las fuentes por separado.

La necesidad de información requerida en este escenario, consiste en conocer las películas que se exhiben en salas de cines de distintas compañías y sus horarios. Con el fin de enriquecer la información de las carteleras se incorporaron los datos de repositorios de datos especializados en información de este ámbito.

Para el desarrollo de este caso de estudio se utilizó información contenida en las fuentes de datos que se mencionan a continuación:

- ***Critics Round Up*** [5]***:*** Es un portal en Internet especializado en contener reseñas y valoraciones de películas (Véase Figura 6.1).

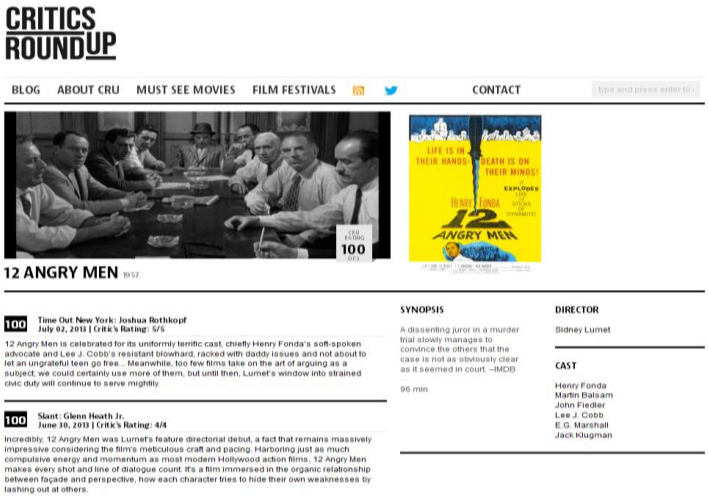


Figura 6.1 Sitio de la base de datos de críticas y resúmenes de películas "Critics Round Up".

La información obtenida de este sitio se resume en los siguientes campos:

- Título de la película.
- Año de estreno.
- Calificación.
- Sinopsis.

- **Internet Movie Database IMDB** [29]**:** Es una base de datos en línea que almacena información relacionada con películas, personal de equipo de producción (incluyendo directores y productores), actores, series de televisión, programas de televisión, entre otros (Véase Figura 6.2).

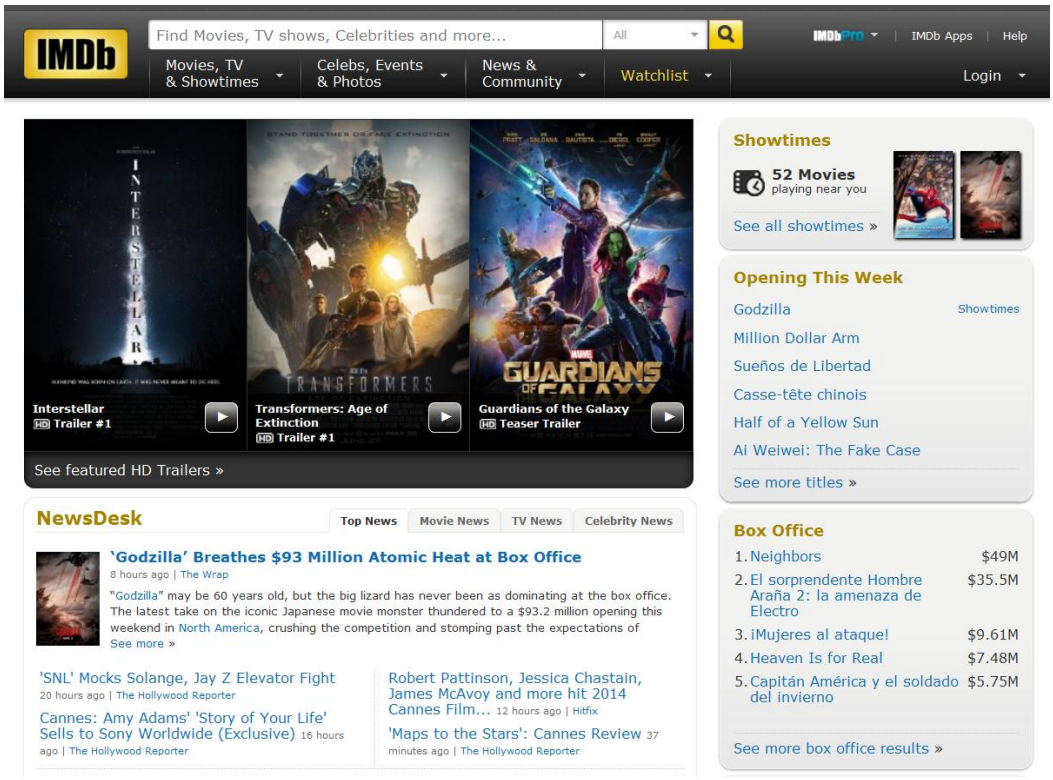


Figura 6.2 Sitio de la base de datos de películas en Internet (Internet Movie Database IMDB).

La información extraída de este sitio consiste en los siguientes campos:

- Título de la película.
- Nombre del protagonista.
- Nombre del director.
- Genero.

- **Carteleras de cines:** La información contenida por las carteleras de los cines es bastante similar, suele incluir el nombre de la película, los distintos horarios a los que se exhibe y recientemente se especifican algunos elementos como lo es el idioma, la clasificación, el género y el tipo de proyección (Véase Figura 6.3).

  Los elementos utilizados para el caso de estudio son los siguientes:

  - Título de la película.
  - El nombre del cine.
  - La hora en la que inicia su función.

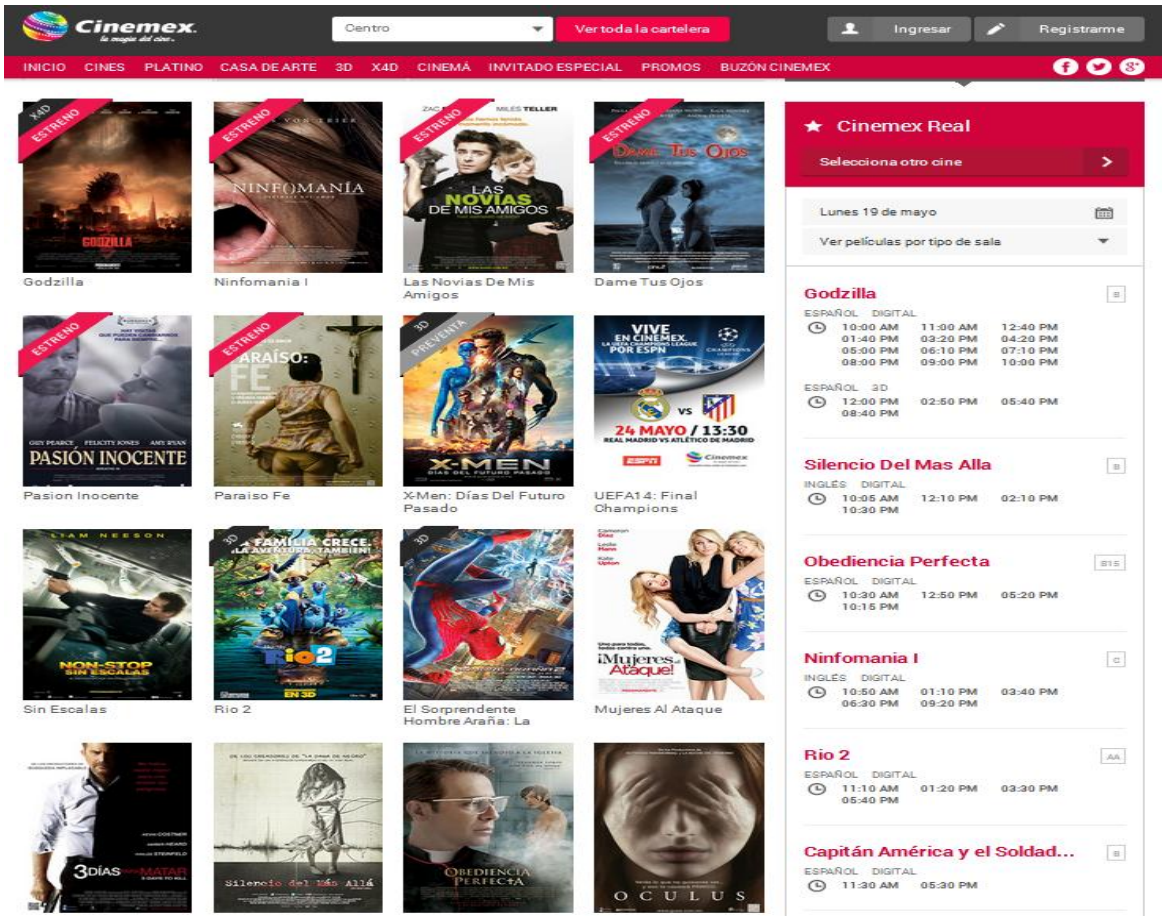


Figura 6.3 Ejemplo de la cartelera de un cine.

6.2.1. Esquema mediado utilizado.

El esquema mediado empleado para el caso de estudio está inspirado en el descrito por Alon Y. Halevy en [30], el cual está conformado por la información provista de nueve bases de datos independientes que almacenan información sobre películas (Véase Figura 6.4).

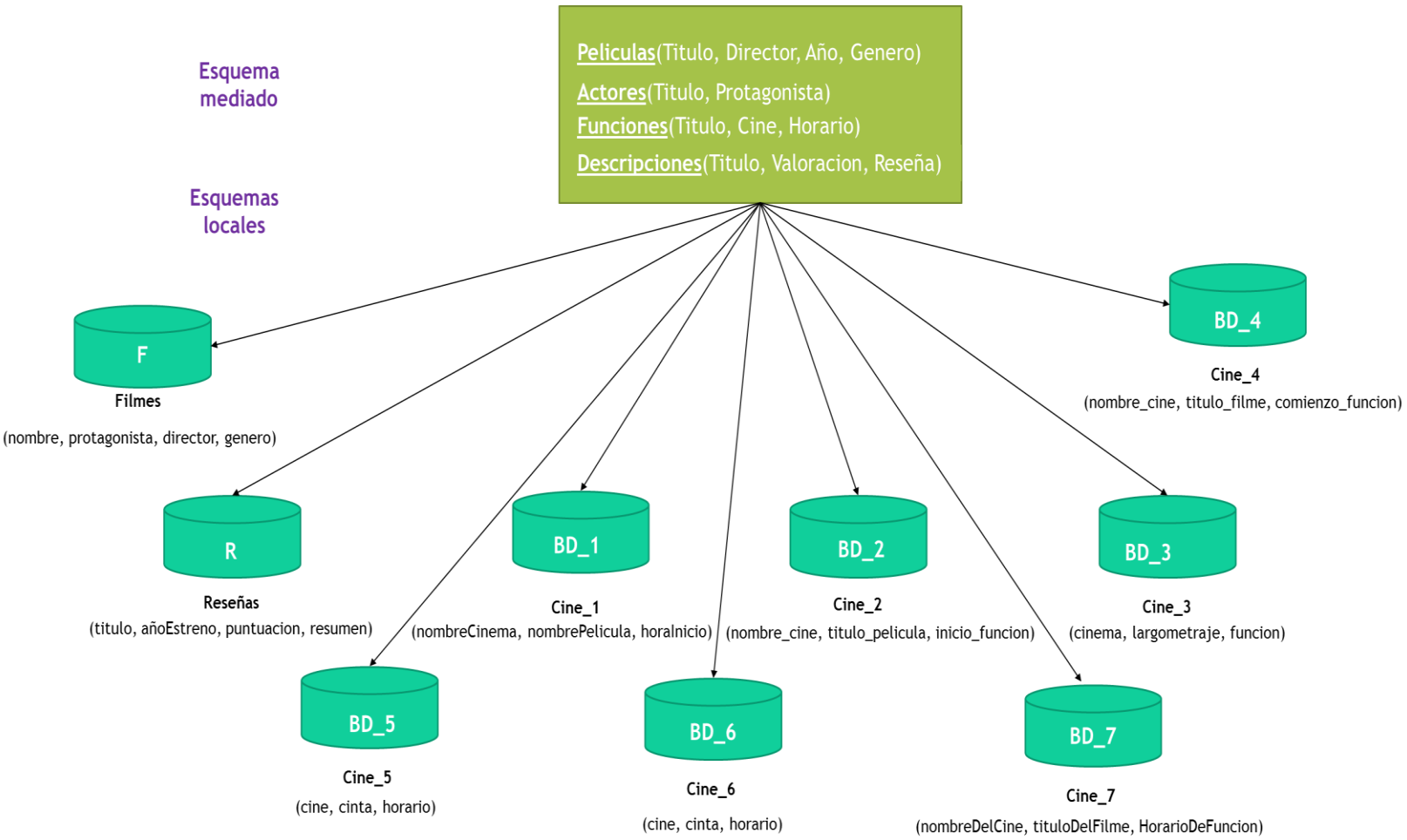


Figura 6.4 Representación del esquema mediado del caso de estudio 1.

A continuación se describen los esquemas de las fuentes de datos autónomas.

La base de datos **F** incluye:

- La tabla **Filmes** contiene los campos:
  - **Nombre:** Es el nombre de la película.
  - **Protagonista:** Es el nombre del actor que tiene el papel principal en la película.
  - **Director:** Es el nombre del director de la película.
  - **Género:** Es la categoría de la película.

La base de datos **R** incluye:

- La tabla **Reseñas** alberga los campos:
  - **Título:** Es el nombre de la película.
  - **AñoEstreno:** Es el año en que se estrenó la película.
  - **Puntuación:** Es la calificación obtenida por la audiencia de la película (esta va del 0 al 100).
  - **Resumen:** Es la sinopsis de la película en la que se describe la temática de la película de modo breve.

Las siguientes siete bases de datos almacenan las carteleras de distintos cines, estas se encuentran constituidas de la siguiente manera:

La base de datos BD_1 incluye:

- La tabla **Cine_1**, que a su vez incluye a los campos:
  - **NombreCinema:** Es el nombre del cine.
  - **NombrePelicula:** Es el nombre de la película.
  - **Inicio_Funcion:** Es el horario en el que inicia la función.

La base de datos BD_2 incluye:

- La tabla **Cine_2**, contiene a los campos:
  - **Nombre_Cine:** Es el nombre del cine.
  - **Titulo_Pelicula:** Es el nombre de la película.
  - **Inicio_Funcion:** Es el horario en el que inicia la función.

La base de datos BD_3 incluye:

- La tabla **Cine_3**, alberga los campos:
  - **Cinema:** Es el nombre del cine.
  - **Largometraje:** Es el nombre de la película.
  - **Función:** Es el horario en el que inicia la función.

La base de datos BD_4 incluye:

- La tabla **Cine_4**, contiene los siguientes campos:
  - **Nombre_Cine:** Es el nombre del cine.
  - **Titulo_Filme:** Es el nombre de la película.
  - **Comienzo_Función:** Es el horario en el que inicia la función.

La base de datos BD_5 incluye:

- La tabla **Cine_5**, que a su vez incluye a los campos:
  - **Cine:** Es el nombre del cine.

- **Cinta:** Es el nombre de la película.
- **Horario:** Es el horario en el que inicia la función.

La base de datos BD_6 incluye:

- La tabla **Cine_6**, está compuesta por los campos:
  - **Cine:** Es el nombre del cine.
  - **Cinta:** Es el nombre de la película.
  - **Horario:** Es el horario en el que inicia la función.

La base de datos BD_7 incluye:

- La tabla **Cine_7**, alberga los siguientes campos:
  - **NombreDelCine:** Es el nombre del cine.
  - **TituloDelFilme:** Es el nombre de la película.
  - **HorarioDeFuncion:** Es el horario en el que inicia la función.

A continuación se resumen las características de las bases de datos utilizadas (Véase Tabla 6.1).

Tabla 6.1 Información de las fuentes de datos empleadas en el caso de estudio 1.

Nombre de la base de datos	Nombre de la tabla	Número de registros	Número de atributos
BD_1	Cine_1	234	3
BD_2	Cine_2	835	3
BD_3	Cine_3	586	3
BD_4	Cine_4	379	3
BD_5	Cine_5	341	3
BD_6	Cine_6	238	3
BD_7	Cine_7	282	3
F	Filmes	278	4
R	Reseñas	275	4

En el siguiente apartado se muestran los planes de ejecución obtenidos del prototipo después de procesar distintas consultas en términos del esquema mediado.

6.2.2. Consulta 1.

Si se desea conocer el nombre del director y el titulo de las películas que están en cartelera, de las cuales su función inicia después de las 13:00 horas y su valoración (ante los críticos) es mayor a 80 puntos, entonces la información solicitada por el usuario puede resumirse en:

Campos a visualizar: Peliculas.Titulo, Peliculas.Director

Condiciones:

C_1: Funciones.Horario > "13:00:00"

C_2: Descripciones.Valoracion > 80

Mediante el proceso de descomposición de la consulta mediada, se obtienen las siguientes consultas objetivo (estas pueden ser contestadas por las bases de datos remotas):

- **Q_1:** USE BD_1@ SELECT Cine_1.NombrePelicula FROM BD_1.Cine_1 WHERE BD_1.Cine_1.HoraInicio> '13:00:00;
- **Q_2:** USE BD_2@ SELECT Cine_2.Titulo_Pelicula FROM BD_2.Cine_2 WHERE BD_2.Cine_2. Inicio_Funcion> '13:00:00';
- **Q_3:** USE BD_3@ SELECT Cine_3.Largometraje FROM BD_3.Cine_3 WHERE BD_3.Cine_3.Funcion> '13:00:00';
- **Q_4:** USE BD_4@ SELECT Cine_4.Titulo_Filme FROM BD_4.Cine_4 WHERE BD_4.Cine_4.Comienzo_Funcion> '13:00:00';
- **Q_5:** USE BD_5@ SELECT Cine_5.Cinta FROM BD_5.Cine_5 WHERE BD_5.Cine_5.Horario> '13:00:00';
- **Q_6:** USE BD_6@ SELECT Cine_6.Cinta FROM BD_6.Cine_6 WHERE BD_6.Cine_6.Horario> '13:00:00';
- **Q_7:** USE BD_7@ SELECT Cine_7.TituloDelFilme FROM BD_7.Cine_7 WHERE BD_7.Cine_7. HorarioDeFuncion > '13:00:00';
- **Q_8:** USE F@ SELECT Filmes.Nombre, F.Filmes.Director FROM F.Filmes;
- **Q_9:** USE R@ SELECT Reseñas.Titulo FROM R.Reseñas WHERE R.Reseñas.Puntuacion > 80 ;

Después de obtener las consultas objetivo, el módulo procede a buscar el orden en el que llevará a cabo la combinación de los resultados de estas consultas.

Con este fin se realizan las estimaciones del número de tuplas que resultarán de la ejecución de las consultas objetivo en sus respectivos sitios, podría suponerse que el número de registros obtenidos por cada fuente es igual al tamaño de la relación; sin embargo, después de aplicar las condiciones de filtrado este número se reduce. Por ello se aplican un conjunto de fórmulas para la

estimación del número de tuplas resultantes de una cláusula de filtrado (Véase Tabla 3.4), de estas se obtienen los siguientes datos (Véase Tabla 6.2).

Tabla 6.2 .Estimación del número de tuplas de los resultados de las consultas remotas.

Consulta remota	Fuente destino	Tabla destino	Tamaño original	Tamaño estimado después de aplicar las condiciones de filtrado	
				Estimado	Real
Q_1	BD_1	Cine_1	234	117	192
Q_2	BD_2	Cine_2	835	417.5	561
Q_3	BD_3	Cine_3	586	293	318
Q_4	BD_4	Cine_4	379	189.5	305
Q_5	BD_5	Cine_5	341	170.5	197
Q_6	BD_6	Cine_6	238	119	158
Q_7	BD_7	Cine_7	282	141	139
Q_8	F	Filmes	278	278	278
Q_9	R	Reseñas	275	99.30	248

El factor en el que disminuye el número de tuplas se debe a razones como el tipo de condición de filtrado y la propia distribución de los datos.

Posteriormente se realizan las estimaciones de los resultados intermedios de las operaciones reunión implicadas en la consulta, para ello se toman en cuenta las consultas objetivo, así como las correspondencias existentes que las relacionan. En la siguiente tabla (Véase Tabla 6.3) se visualizan algunos de los resultados obtenidos:

Tabla 6.3 Estimación del número de tuplas de los resultados intermedios.

Correspondencia	Base 1	Base 2	Número calculado de tuplas resultantes
ID="13"	bd_1.cine_1.nombrePelicula	bd_2.cine_2.titulo_pelicula	1135
ID="16"	bd_3.cine_3.largometraje	bd_2.cine_2.titulo_pelicula	2844
ID="19"	bd_3.cine_3.largometraje	bd_4.cine_4.titulo_filme	1542

ID="21"	bd_1.cine_1.nombrePelicula	bd_3.cine_3.largometraje	952
ID="23"	bd_2.cine_2.nombre_cine	bd_4.cine_4.nombre_cine	2825
ID="24"	bd_2.cine_2.titulo_pelicula	bd_4.cine_4.titulo_filme	1839
ID="27"	bd_3.cine_3.largometraje	bd_5.cine_5.cinta	1387
ID="30"	bd_4.cine_4.titulo_filme	bd_6.cine_6.cinta	593
ID="35"	bd_6.cine_6.cinta	f.filmes.nombre	119
ID="36"	bd_7.cine_7.tituloDelFilme	r.reseñas.titulo	141
ID="38"	bd_1.cine_1.nombrePelicula	bd_4.cine_4.titulo_filme	739
ID="41"	bd_2.cine_2.titulo_pelicula	bd_5.cine_5.cinta	1655
ID="44"	bd_3.cine_3.largometraje	bd_6.cine_6.cinta	917
ID="47"	bd_4.cine_4.titulo_filme	bd_7.cine_7.tituloDelFilme	703
ID="49"	bd_5.cine_5.cinta	f.filmes.nombre	170
ID="50"	bd_6.cine_6.cinta	r.reseñas.titulo	119
ID="52"	bd_1.cine_1.nombrePelicula	bd_5.cine_5.cinta	569
ID="55"	bd_2.cine_2.titulo_pelicula	bd_6.cine_6.cinta	1155
ID="58"	bd_3.cine_3.largometraje	bd_7.cine_7.tituloDelFilme	1087
ID="60"	bd_4.cine_4.titulo_filme	f.filmes.nombre	189
ID="61"	bd_5.cine_5.cinta	r.reseñas.titulo	170
ID="63"	bd_1.cine_1.nombrePelicula	bd_6.cine_6.cinta	366
ID="66"	bd_2.cine_2.titulo_pelicula	bd_7.cine_7.tituloDelFilme	1369
ID="68"	bd_3.cine_3.largometraje	f.filmes.nombre	293
ID="69"	bd_4.cine_4.titulo_filme	r.reseñas.titulo	189
ID="71"	bd_1.cine_1.nombrePelicula	bd_7.cine_7.tituloDelFilme	434
ID="73"	bd_2.cine_2.titulo_pelicula	f.filmes.nombre	417
ID="74"	bd_3.cine_3.largometraje	r.reseñas.titulo	293
ID="75"	bd_1.cine_1.nombrePelicula	f.filmes.nombre	117

ID="76"	bd_2.cine_2.titulo_pelicula	r.reseñas.titulo	417
ID="77"	bd_1.cine_1.nombrePelicula	r.reseñas.titulo	117

Una vez que se tiene una estimación del número de tuplas que resultarán de las reuniones, es posible establecer una prioridad en su ejecución, que como se ha mencionado anteriormente tiene como objetivo omitir el procesamiento innecesario de tuplas o al menos postergarlo hasta que sea inevitable, para ello se utiliza un algoritmo Greedy que toma como entrada a las consultas objetivo, las correspondencias definidas y los datos previamente calculados, este en la primera iteración considera como mejor opción realizar la combinación de los datos obtenidos de las consultas Q_1 y Q_9 (en este apartado el termino Q_n consiste en los resultados obtenidos de la consulta local n) utilizando como condición de reunión los campos que hacen referencia al nombre de la película (esta tiene el número setenta y siete de la anterior tabla), debido a que se vislumbra que el llevar a cabo esta reunión se generará la menor cantidad de tuplas, por consiguiente en la siguiente operación de reunión el operando que manejará será el más pequeño posible, por lo que se asume que conllevará a un menor procesamiento y por consiguiente a un menor tiempo de respuesta.

Posteriormente, el algoritmo busca de entre todas las opciones restantes (Q_1, Q_2, Q_3, Q_4, Q_5, Q_7, Q_8) la opción que genere el resultado menor al reunirse con los datos provistos por la reunión que se realizó previamente (Q_1, Q_9), esto según los datos analizados ocurre mediante la reunión del resultado provisional con Q_8 (usando la correspondencia de reunión setenta y cinco).

Este proceso itera hasta que se ha definido una prioridad para cada una de las consultas objetivo, al concluir este paso se obtiene un orden de ejecución para realizar las combinaciones de los datos obtenidos de las fuentes remotas, esta información se resume en la siguiente tabla (Véase Tabla 6.4), en ella se resume el orden los elementos que se reúnen(usando la notación Join(A,B)), la condición para realizar la reunión y el número de tuplas que se obtuvieron con y sin repetidos(mediante la utilización del operador DISTINCT sobre los resultados obtenidos).

Tabla 6.4 Plan de ejecución generado por el prototipo para la consulta 1.

Orden de ejecución	Operación	Condición de reunión	Número de tuplas resultantes	
			Con repetidos	Sin repetidos
1	Q_a=Join(Q_1,Q_9)	77	64	13
2	Q_b=Join(Q_a,Q_8)	75	64	13
3	Q_c=Join(Q_b,Q_6)	50	199	10

4	Q_d=Join(Q_c,Q_7)	36	1318	9
5	Q_e=Join(Q_d,Q_5)	61	8094	8
6	Q_f=Join(Q_e,Q_4)	69	232500	8
7	Q_g=Join(Q_f,Q_3)	74	3251516	8
8	Q_h=Join(Q_g,Q_2)	76	182076119	8

Con el fin de realizar una comparación objetiva de la eficiencia del plan de ejecución desarrollado se utilizaron los planes generados por el optimizador del sistema de administración de base de datos MySQL como punto de comparación, en la siguiente tabla (Véase Tabla 6.5) puede verse el número de tuplas generado en cada reunión de dicho plan.

Tabla 6.5 Plan de ejecución generado por el MySQL para la consulta 1.

Orden de ejecución	Operación	Número de tuplas resultantes	
		Con repetidos	Sin repetidos
1	Q_a=Join(Q_1,Q_6)	579	16
2	Q_b=Join(Q_a,Q_9)	199	10
3	Q_c=Join(Q_b,Q_8)	199	10
4	Q_d=Join(Q_c,Q_7)	1318	9
5	Q_e=Join(Q_d,Q_5)	8094	8
6	Q_f=Join(Q_e,Q_4)	232500	8
7	Q_g=Join(Q_f,Q_3)	3251516	8
8	Q_h=Join(Q_g,Q_2)	182076119	8

En la Figura 6.5 se muestra una comparativa del número de tuplas procesadas por cada operación de reunión en el plan generado por el prototipo con respecto a las tuplas procesadas por el plan de ejecución brindado por MySQL, en esta se puede ver que el plan de ejecución desarrollado por el prototipo en la primera reunión procesa 500 tuplas menos que su contra parte, en la siguiente reunión hay una diferencia de poco más de 100 tuplas, sin embargo, para la tercera hasta la octava reunión se procesan la misma cantidad de tuplas.

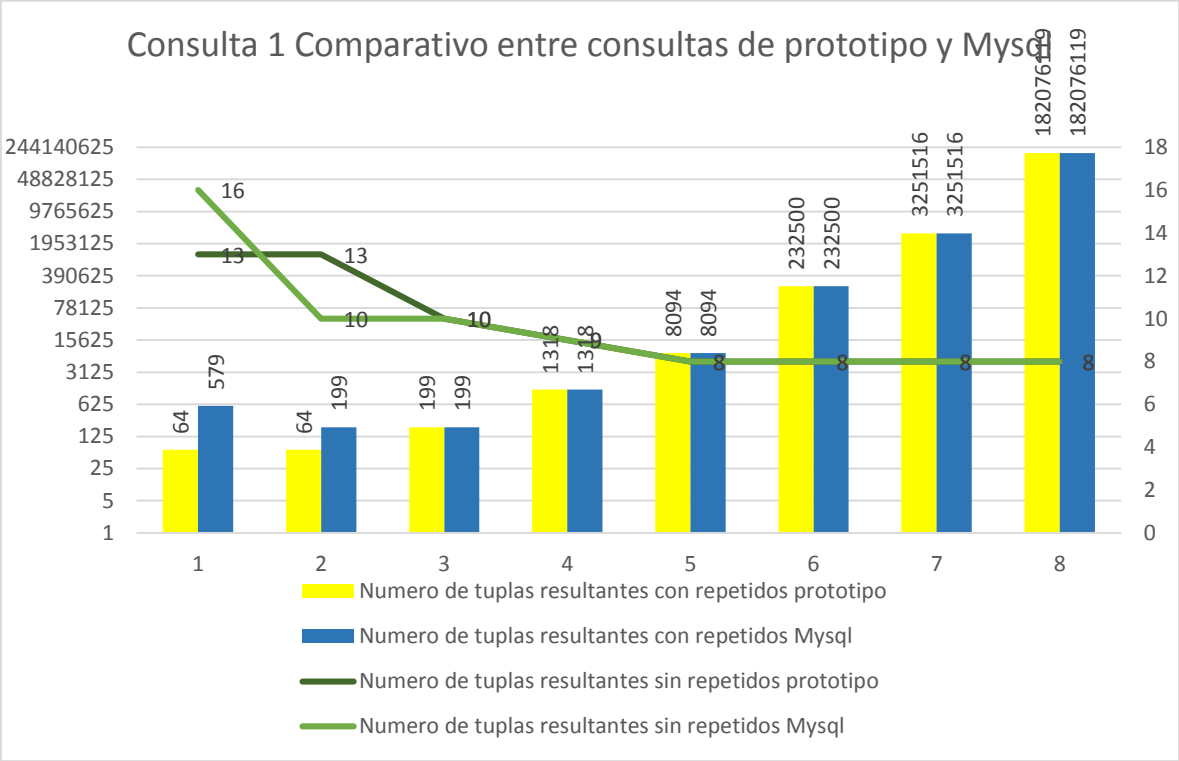


Figura 6.5 Comparativa de las tuplas procesadas en el plan generado por el prototipo y el de MySQL

6.2.3. Consulta 2.

Supóngase que se desea conocer el género, el nombre del director y el nombre de las películas cuyo año de estreno este en el intervalo de 1994 y 2012, además que su hora de inicio de función este entre las 11:00 y 18:00 horas.

La información anteriormente descrita puede sintetizarse de la siguiente manera:

Campos a visualizar: Peliculas.Titulo, Peliculas.Director, Peliculas.Genero

Condiciones:

C_1: Funciones.Horario > “11:00:00”

C_2: Funciones.Horario < “18:00:00”

C_3: Peliculas.Año > 1994

C_4: Peliculas.Año < 2012

Mediante el proceso de descomposición de la consulta mediada, se obtienen las siguientes consultas objetivo:

- **Q_1:** USE BD_1@ SELECT Cine_1.NombrePelicula FROM BD_1.Cine_1 WHERE BD_1.Cine_1.HoraInicio > '11:00:00' AND BD_1.Cine_1.HoraInicio < '18:00:00';
- **Q_2:** USE BD_2@ SELECT Cine_2.Titulo_Pelicula FROM BD_2.Cine_2 WHERE BD_2.Cine_2. Inicio_Funcion > '11:00' AND BD_2.Cine_2. Inicio_Funcion < '18:00:00';
- **Q_3:** USE BD_3@ SELECT Cine_3.Largometraje FROM BD_3.Cine_3 WHERE BD_3.Cine_3.Funcion > '11:00:00' AND BD_3.Cine_3.Funcion < '18:00:00';
- **Q_4:** USE BD_4@ SELECT Cine_4.Titulo_Filme FROM BD_4.Cine_4 WHERE BD_4.Cine_4.Comienzo_Funcion> '11:00:00' AND BD_4.Cine_4.Comienzo_Funcion < '18:00:00';
- **Q_5:** USE BD_5@ SELECT Cine_5.Cinta FROM BD_5.Cine_5 WHERE BD_5.Cine_5.Horario> '11:00:00' AND BD_5.Cine_5.Horario< '18:00:00';
- **Q_6:** USE BD_6@ SELECT Cine_6.Cinta FROM BD_6.Cine_6 WHERE BD_6.Cine_6.Horario> '11:00:00' AND BD_6.Cine_6.Horario < '18:00:00';
- **Q_7:** USE BD_7@ SELECT Cine_7.TituloDelFilme FROM BD_7.Cine_7 WHERE BD_7.Cine_7.HorarioDeFuncion > '11:00:00' AND BD_7.Cine_7.HorarioDeFuncion < '18:00:00';
- **Q_8**: USE F@ SELECT F.Filmes.Nombre, F.Filmes.Director, F.Filmes.Genero FROM F.Filmes;
- **Q_9**: USE R@ SELECT Reseñas.Titulo FROM R.Reseñas WHERE R.Reseñas.añoEstreno > 1994 AND R.Reseñas.añoEstreno > 2012;

Después de obtener las consultas objetivo, el módulo procede a buscar el orden en el que llevará a cabo la combinación de los resultados de estas consultas, de la misma manera a como se hizo en el ejemplo anterior, se procede a realizar las estimaciones del número de tuplas que resultarán de la ejecución de las consultas objetivo en las fuentes de datos remotas; pero ahora también se considera la evaluación de dos intervalos, por ello se realizan los cálculos pertinentes para estimar el número de tuplas resultantes de la conjunción de dos predicados (Véase Figura 3.6), la información obtenida se muestra en la siguiente tabla (Véase Tabla 6.6).

Tabla 6.6 .Estimación del número de tuplas de los resultados de las consultas remotas.

Consulta remota	Fuente destino	Tabla destino	Tamaño original	Tamaño estimado después de aplicar las condiciones de filtrado	
				Estimado	Real
Q_1	BD_1	Cine_1	234	58.5	158
Q_2	BD_2	Cine_2	835	208.75	792

Q_3	BD_3	Cine_3	586	146.5	515
Q_4	BD_4	Cine_4	379	94.75	349
Q_5	BD_5	Cine_5	341	85.25	302
Q_6	BD_6	Cine_6	238	59.5	208
Q_7	BD_7	Cine_7	282	70.5	224
Q_8	F	Filmes	278	278	278
Q_9	R	Reseñas	275	57.86	14

De los resultados obtenidos, es posible percatarse que la existencia de dos consultas conjuntivas sobre un mismo campo promete reducir el número de tuplas obtenidas de las fuentes de datos, en algunas fuentes como Q_9 se puede percibir de manera más evidente, que en otras como en Q_2. A pesar de esta situación es posible identificar algunos aspectos importantes, como que la consulta Q_9 brindará el resultado de tamaño menor, que Q_1 es la segunda o que Q_6 es la tercera mejor opción.

Posteriormente se realizan los cálculos pertinentes para obtener las estimaciones de los resultados intermedios de las operaciones de reunión implicadas en la consulta (tomando en cuenta las consultas objetivo y las correspondencias que las relacionan). Los resultados obtenidos de estos cálculos se proporcionan en la siguiente tabla (Véase Tabla 6.7).

Tabla 6.7 Estimación del número de tuplas de los resultados intermedios.

Correspondencia	Base 1	Base 2	Número calculado de tuplas resultantes
ID="13"	bd_1.cine_1.nombrePelicula	bd_2.cine_2.titulo_pelicula	283
ID="16"	bd_3.cine_3.largometraje	bd_2.cine_2.titulo_pelicula	711
ID="19"	bd_3.cine_3.largometraje	bd_4.cine_4.titulo_filme	385
ID="21"	bd_1.cine_1.nombrePelicula	bd_3.cine_3.largometraje	238
ID="23"	bd_2.cine_2.nombre_cine	bd_4.cine_4.nombre_cine	706

ID="24"	bd_2.cine_2.titulo_pelicula	bd_4.cine_4.titulo_filme	459
ID="27"	bd_3.cine_3.largometraje	bd_5.cine_5.cinta	346
ID="30"	bd_4.cine_4.titulo_filme	bd_6.cine_6.cinta	148
ID="35"	bd_6.cine_6.cinta	f.filmes.nombre	59
ID="36"	bd_7.cine_7.tituloDelFilme	r.reseñas.titulo	70
ID="38"	bd_1.cine_1.nombrePelicula	bd_4.cine_4.titulo_filme	184
ID="41"	bd_2.cine_2.titulo_pelicula	bd_5.cine_5.cinta	413
ID="44"	bd_3.cine_3.largometraje	bd_6.cine_6.cinta	229
ID="47"	bd_4.cine_4.titulo_filme	bd_7.cine_7.tituloDelFilme	175
ID="49"	bd_5.cine_5.cinta	f.filmes.nombre	85
ID="50"	bd_6.cine_6.cinta	r.reseñas.titulo	59
ID="52"	bd_1.cine_1.nombrePelicula	bd_5.cine_5.cinta	142
ID="55"	bd_2.cine_2.titulo_pelicula	bd_6.cine_6.cinta	288
ID="58"	bd_3.cine_3.largometraje	bd_7.cine_7.tituloDelFilme	271
ID="60"	bd_4.cine_4.titulo_filme	f.filmes.nombre	94
ID="61"	bd_5.cine_5.cinta	r.reseñas.titulo	85
ID="63"	bd_1.cine_1.nombrePelicula	bd_6.cine_6.cinta	91
ID="66"	bd_2.cine_2.titulo_pelicula	bd_7.cine_7.tituloDelFilme	342

ID="68"	bd_3.cine_3.largometraje	f.filmes.nombre	146
ID="69"	bd_4.cine_4.titulo_filme	r.reseñas.titulo	94
ID="71"	bd_1.cine_1.nombrePelicula	bd_7.cine_7.tituloDelFilme	108
ID="73"	bd_2.cine_2.titulo_pelicula	f.filmes.nombre	208
ID="74"	bd_3.cine_3.largometraje	r.reseñas.titulo	146
ID="75"	bd_1.cine_1.nombrePelicula	f.filmes.nombre	58
ID="76"	bd_2.cine_2.titulo_pelicula	r.reseñas.titulo	208
ID="77"	bd_1.cine_1.nombrePelicula	r.reseñas.titulo	58

Una vez obtenidas las estimaciones de los tamaños intermedios, se ejecuta el algoritmo para establecer un orden de ejecución para realizar las combinaciones de los datos obtenidos de las fuentes remotas, en la siguiente tabla (VéaseTabla 6.8) se muestra el plan de ejecución generado.

Tabla 6.8 Plan de ejecución generado por el prototipo para la consulta 2.

Orden de ejecución	Operación	Condición de reunión	Número de tuplas resultantes	
			Con repetidos	Sin repetidos
1	Q_a=Join(Q_1,Q_8)	75	158	25
2	Q_b=Join(Qa,Q_9)	77	86	16
3	Q_c=Join(Q_b,Q_6)	35	456	11
4	Q_d=Join(Q_c,Q_7)	36	2301	11
5	Q_e=Join(Q_d,Q_5)	49	30280	10
6	Q_f=Join(Q_e,Q_4)	60	517858	10
7	Q_g=Join(Q_e,Q_3)	68	14571394	10

8	Q_h=Join(Q_e,Q_2)	73	436574177	9

Nuevamente con el fin de manejar un punto de comparación, en la siguiente tabla (Véase Tabla 6.9) se presenta el histórico del número de tuplas procesadas por el plan de ejecución obtenido de MySQL para la misma consulta.

Tabla 6.9 Plan de ejecución generado por el MySQL para la consulta 2.

Orden de ejecución	Operación	Numero de tuplas resultantes	
		Con repetidos	Sin repetidos
1	Q_a=Join(Q_1,Q_6)	555	15
2	Q_b=Join(Q_a,Q_9)	456	11
3	Q_c=Join(Q_b,Q_8)	465	11
4	Q_d=Join(Q_c,Q_7)	2301	11
5	Q_e=Join(Q_d,Q_5)	30280	10
6	Q_f=Join(Q_e,Q_4)	517858	10
7	Q_g=Join(Q_f,Q_3)	14571394	10
8	Q_h=Join(Q_g,Q_2)	436574177	9

En la siguiente figura (Véase Figura 6.6) se muestra una comparativa del número de tuplas procesadas en cada reunión del plan generado mediante la aproximación propuesta con respecto a las tuplas procesadas por el plan de ejecución brindado por MySQL, en esta se puede ver que mediante la ejecución del plan desarrollado por el prototipo en la primer reunión se ejecutan 350 tuplas menos que su contraparte, en la segunda reunión se procesan casi 400 tuplas, sin embargo de la tercera reunión hasta la octava, se procesan el mismo número de tuplas por cada reunión del plan de ejecución.

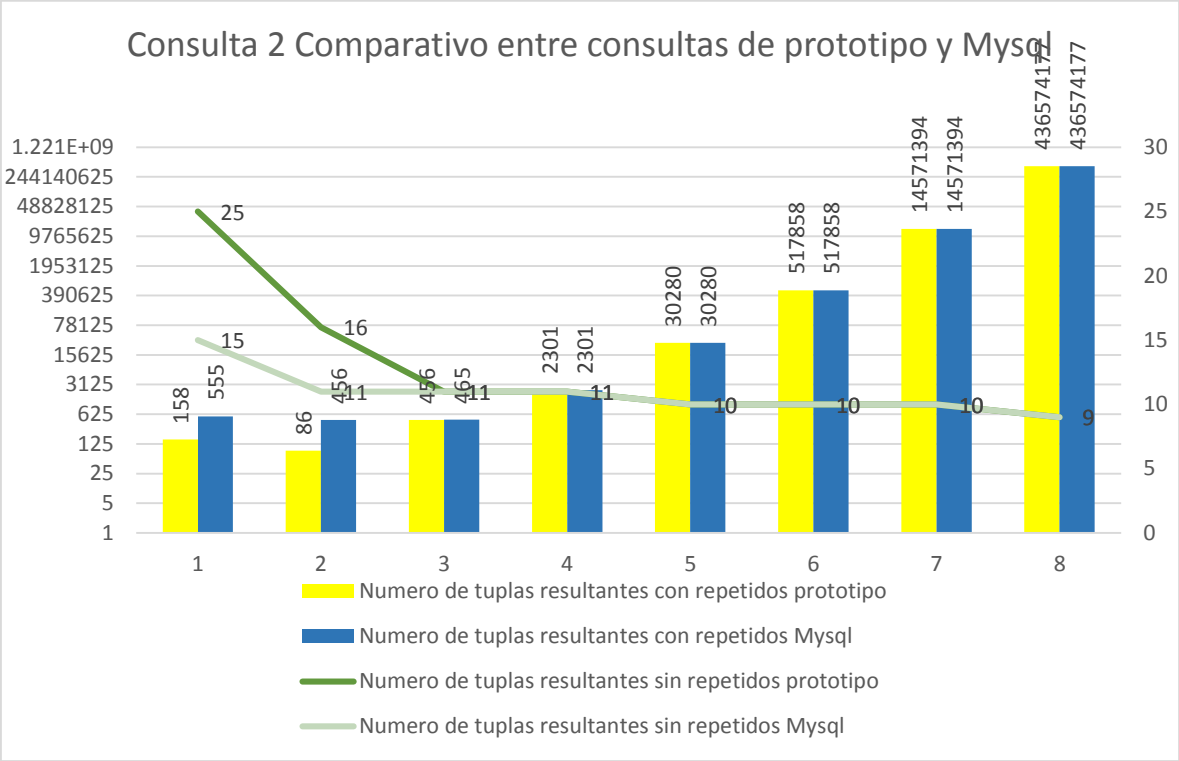


Figura 6.6 Comparativa de las tuplas procesadas en el plan generado por el prototipo y el de MySQL

6.2.3. Consulta 3.

Suponga que se desea conocer el nombre y reseña de las películas cuyo horario de función sea después de las 14:00 horas, esta necesidad de información sintetizarse de la siguiente manera:

Campos a visualizar: Funciones.Titulo

Condiciones:

C_1: Funciones.Horario > 14:00:00

Mediante el proceso de descomposición de la consulta mediada, se obtienen las siguientes consultas objetivo:

- **Q_1:** USE BD_1@ SELECT Cine_1.NombrePelicula FROM BD_1.Cine_1 WHERE BD_1.Cine_1.HoraInicio> '14:00:00';
- **Q_2:** USE BD_2@ SELECT Cine_2.Titulo_Pelicula FROM BD_2.Cine_2 WHERE BD_2.Cine_2. Inicio_Funcion> '14:00:00';
- **Q_3:** USE BD_3@ SELECT Cine_3.Largometraje FROM BD_3.Cine_3 WHERE BD_3.Cine_3.Funcion> '14:00:00';

- **Q_4:** USE BD_4@ SELECT Cine_4.Titulo_Filme FROM BD_4.Cine_4 WHERE BD_4.Cine_4.Comienzo_Funcion> '14:00:00';
- **Q_5:** USE BD_5@ SELECT Cine_5.Cinta FROM BD_5.Cine_5 WHERE BD_5.Cine_5.Horario> '14:00:00';
- **Q_6:** USE BD_6@ SELECT Cine_6.Cinta FROM BD_6.Cine_6 WHERE BD_6.Cine_6.Horario> '14:00:00';
- **Q_7:** USE BD_7@ SELECT Cine_7.TituloDelFilme FROM BD_7.Cine_7 WHERE BD_7.Cine_7.HorarioDeFuncion > '14:00:00';

En este caso, no se establece alguna condición de filtrado que involucre a la información complementaria de las carteleras (a diferencia de la anterior consulta), por ello las consultas objetivo consideradas únicamente están destinadas a las fuentes que albergan las carteleras de los cines, después de obtener las consultas objetivo, el módulo procede a buscar el orden en el que llevará a cabo la combinación de los resultados de estas consultas, de la misma manera como se realizó en el ejemplo anterior se realizan las estimaciones del número de tuplas que resultarán de la ejecución de las consultas objetivo en las fuentes de datos remotas, la información obtenida se muestra en (Véase Tabla 6.10).

Tabla 6.10 .Estimación del número de tuplas de los resultados de las consultas remotas.

Consulta remota	Fuente destino	Tabla destino	Tamaño original	Tamaño estimado después de aplicar las condiciones de filtrado	
				Estimado	Real
Q_1	BD_1	Cine_1	234	117	131
Q_2	BD_2	Cine_2	835	417.5	434
Q_3	BD_3	Cine_3	586	293	279
Q_4	BD_4	Cine_4	379	189.5	188
Q_5	BD_5	Cine_5	341	170.5	147
Q_6	BD_6	Cine_6	238	119	135
Q_7	BD_7	Cine_7	282	141	138

Posteriormente, se realizan las estimaciones de los resultados intermedios de las operaciones reunión implicadas en la consulta, en la siguiente tabla (Véase Tabla 6.11) se visualizan algunos de los resultados obtenidos.

Tabla 6.11 Estimación del número de tuplas de los resultados intermedios.

Correspondencia	Base 1	Base 2	Número calculado de tuplas resultantes
ID="13"	bd_1.cine_1.nombrePelicula	bd_2.cine_2.titulo_pelicula	1135
ID="16"	bd_3.cine_3.largometraje	bd_2.cine_2.titulo_pelicula	2844
ID="19"	bd_3.cine_3.largometraje	bd_4.cine_4.titulo_filme	1542
ID="21"	bd_1.cine_1.nombrePelicula	bd_3.cine_3.largometraje	952
ID="23"	bd_2.cine_2.nombre_cine	bd_4.cine_4.nombre_cine	2825
ID="24"	bd_2.cine_2.titulo_pelicula	bd_4.cine_4.titulo_filme	1839
ID="27"	bd_3.cine_3.largometraje	bd_5.cine_5.cinta	1387
ID="30"	bd_4.cine_4.titulo_filme	bd_6.cine_6.cinta	593
ID="35"	bd_6.cine_6.cinta	f.filmes.nombre	-------
ID="36"	bd_7.cine_7.tituloDelFilme	r.reseñas.titulo	-------
ID="38"	bd_1.cine_1.nombrePelicula	bd_4.cine_4.titulo_filme	739
ID="41"	bd_2.cine_2.titulo_pelicula	bd_5.cine_5.cinta	1655
ID="44"	bd_3.cine_3.largometraje	bd_6.cine_6.cinta	917
ID="47"	bd_4.cine_4.titulo_filme	bd_7.cine_7.tituloDelFilme	703
ID="49"	bd_5.cine_5.cinta	f.filmes.nombre	-------
ID="50"	bd_6.cine_6.cinta	r.reseñas.titulo	-------

ID="52"	bd_1.cine_1.nombrePelicula	bd_5.cine_5.cinta	569
ID="55"	bd_2.cine_2.titulo_pelicula	bd_6.cine_6.cinta	1155
ID="58"	bd_3.cine_3.largometraje	bd_7.cine_7.tituloDelFilme	1087
ID="60"	bd_4.cine_4.titulo_filme	f.filmes.nombre	-------
ID="61"	bd_5.cine_5.cinta	r.reseñas.titulo	-------
ID="63"	bd_1.cine_1.nombrePelicula	bd_6.cine_6.cinta	366
ID="66"	bd_2.cine_2.titulo_pelicula	bd_7.cine_7.tituloDelFilme	1369
ID="68"	bd_3.cine_3.largometraje	f.filmes.nombre	-------
ID="69"	bd_4.cine_4.titulo_filme	r.reseñas.titulo	-------
ID="71"	bd_1.cine_1.nombrePelicula	bd_7.cine_7.tituloDelFilme	434
ID="73"	bd_2.cine_2.titulo_pelicula	f.filmes.nombre	-------
ID="74"	bd_3.cine_3.largometraje	r.reseñas.titulo	-------
ID="75"	bd_1.cine_1.nombrePelicula	f.filmes.nombre	-------
ID="76"	bd_2.cine_2.titulo_pelicula	r.reseñas.titulo	-------
ID="77"	bd_1.cine_1.nombrePelicula	r.reseñas.titulo	-------

Después se ejecuta el algoritmo para determinar el orden de ejecución en el que se realizarán las combinaciones de los datos obtenidos de las fuentes remotas, en la siguiente tabla (Véase Tabla 6.12) se puede ver el plan de ejecución obtenido.

Tabla 6.12 Plan de ejecución generado para la consulta 3.

Orden de ejecución	Operación	Condición de reunión	Número de tuplas resultantes	
			Con repetidos	Sin repetidos
1	Q_a=Join(Q_1,Q_6)	63	264	10
2	Q_b=Join(Qa,Q_7)	71	2904	12
3	Q_c=Join(Q_b,Q_5)	52	8008	11
4	Q_d=Join(Q_c,Q_4)	30	232232	5
5	Q_e=Join(Q_d,Q_3)	44	3251248	3
6	Q_f=Join(Q_e,Q_2)	13	182069888	2

A continuación en (Véase Tabla 6.13) se exhibe el número de tuplas procesadas por cada reunión usando el plan de ejecución generado por el sistema de administración de bases de datos MySQL.

Tabla 6.13 Plan de ejecución generado por el MySQL para la consulta 3.

Orden de ejecución	Operación	Número de tuplas resultantes	
		Con repetidos	Sin repetidos
1	Q_a=Join(Q_1,Q_6)	264	10
2	Q_b=Join(Q_c,Q_7)	2904	12
3	Q_c=Join(Q_d,Q_5)	8008	11
4	Q_f=Join(Q_e,Q_4)	232232	5
5	Q_g=Join(Q_f,Q_3)	3251248	3
6	Q_h=Join(Q_g,Q_2)	182069888	2

En este caso el plan generado por el prototipo y el desarrollado por MySQL son idénticos, esto se puede ver en la siguiente figura (Véase Figura 6.7) donde se muestra la comparativa del número de tuplas procesadas por cada reunión del plan generado mediante la aproximación propuesta con respecto a las tuplas procesadas por el plan de ejecución brindado por MySQL.

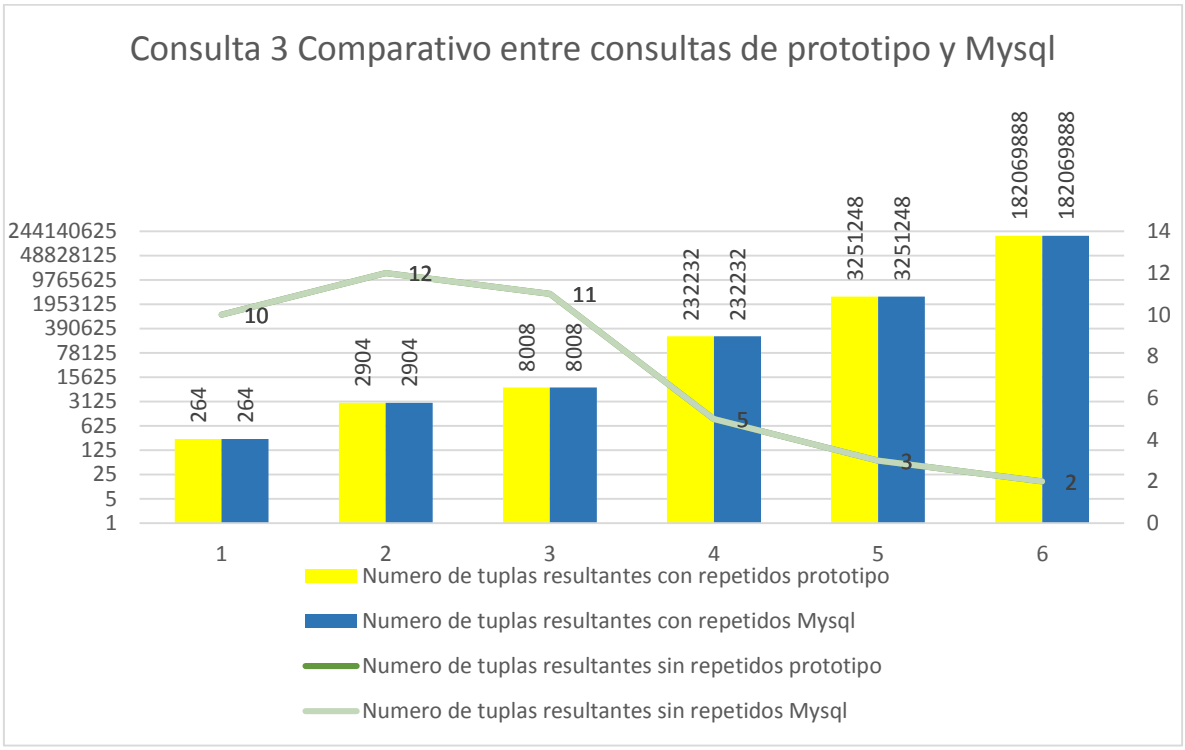


Figura 6.7 Comparativa de las tuplas procesadas en el plan generado por el prototipo y el de MySQL

6.2.4. Consulta 4.

Suponga que se desea conocer tanto el nombre del director así como de la película, de aquellas cintas que pertenecen al género comedia y su año de estreno está comprendido entre los años 2000 y 2014 pero omitiendo a las del año 2009.

Campos a visualizar: Peliculas.Titulo, Peliculas.Director

Condiciones:

C_1: Peliculas.Genero == 'Comedy'

C_2: Peliculas.Año > 2004

C_3: Peliculas.Año < 2014

C_4: Peliculas.Año <> 2009

Como resultado de la etapa de descomposición se obtienen solo dos consultas, esto debido a que en la consulta no se solicitó visualizar ni evaluar ningún campo que involucre la información de carteleras, entonces el prototipo omite considerarlas durante la etapa de generación de consultas objetivo.

- **Q_1:** USE F@ SELECT Filmes.Nombre, Filmes.Director FROM BD_1.Cine_1 WHERE F.Filmes.Genero != 'Comedy';
- **Q_2:** USE R@ SELECT Reseñas.Titulo FROM R.Reseñas WHERE R.Reseñas.añoEstreno > 2004 AND R.Reseñas.añoEstreno < 2012 AND R.Reseñas.añoEstreno != 2009;

Al realizar los cálculos referentes a la evaluación de condiciones de filtrado se obtienen los siguientes resultados (Véase Tabla 6.14).

Tabla 6.14 .Estimación del número de tuplas de los resultados de las consultas remotas.

Consulta remota	Fuente destino	Tabla destino	Tamaño original	Tamaño estimado después de aplicar las condiciones de filtrado	
				Estimado	Real
Q_1	F	Filmes	278	16.33	29
Q_2	R	Reseñas	275	40.86	56

En este caso al haber solo dos consultas objetivo, entonces el programa únicamente busca alguna correspondencia mediante la cual poder llevar a cabo la reunión de dichos datos (Véase Tabla 6.15).

Tabla 6.15 Correspondencia encontrada para realizar la operación de reunión.

Correspondencia	Base 1	Base 2
ID="31"	r.reseñas.titulo	f.filmes.nombre

6.2.5. Consulta 5.

Suponga que se desea conocer el nombre de las películas y sus respectivas reseñas, de aquellas que han obtenido más de 90 puntos por los críticos en el periodo 1980 al 2000, en términos del esquema mediado la necesidad de información podría sintetizarse en la siguiente expresión.

Campos a visualizar: Descripciones.Titulo, Descripciones.Reseña

Condiciones:

C_1: Descripciones.Valoracion > '90'

C_2: Peliculas.Año > 1980

C_3: Peliculas.Año < 2000

Como resultado de la etapa de descomposición se obtiene únicamente una consulta, esto debido

a que los campos que se desean visualizar y sobre los que se especificaron condiciones de filtrado pertenecen únicamente a un sitio.

- **Q_1:** USE R@ SELECT Reseñas.Titulo, Reseñas.Resumen FROM R.Reseñas WHERE R.Reseñas.añoEstreno > 1980 AND R.Reseñas.añoEstreno < 2000 AND R.Reseñas.Puntuacion > 90;

Al tener únicamente una consulta, entonces el plan de ejecución consiste únicamente en transmitir la consulta a la fuente de datos remota.

6.3. Caso de estudio: "Hospitales".

En este apartado se explora la prueba de concepto de la utilización de un caso de estudio en el que se consideran las necesidades del sistema RieSis, este software coordina los servicios de emergencia, rescate, atención en hospitales, registro de víctimas, y seguridad pública en caso de una contingencia severa en la ciudad de México. En este software se integra una base de datos en la que se detalla la ubicación de grúas, hospitales y albergues, así como los recursos humanos disponibles para el envío de ayuda. Debido a la importancia que tiene un sistema de apoyo en caso de desastres, resultaría conveniente que este sistema tuviera acceso total a la información de las distintas dependencias con las que interactúa, suponga que esto no ocurre y que la información que maneja este sistema con respecto al control de hospitales se obtiene empleando un sistema de integración virtual, mediante un acceso limitado a cuatro fuentes de datos distintas, a continuación se muestran los esquemas de dichas fuentes:

Hospitales_1 contiene los siguientes campos:

- **Ident:** Es la clave que identifica un hospital.
- **NomHosp:** Es el nombre del hospital.
- **idDirector:** Consiste en la clave que identifica al administrador del hospital.

Hospitales_2 contiene los siguientes campos:

- **Clave:** Es la clave que identifica un hospital.
- **DirH:** Es la dirección del hospital.
- **latH:** Consiste en la latitud del punto geográfico donde se encuentra ubicado el hospital.
- **longH:** Consiste en la longitud del punto geográfico donde se encuentra ubicado el hospital.

Hospitales_3 contiene los siguientes campos:

- **Id:** Es la clave que identifica un hospital.
- **Clinica:** Es el nombre del hospital.
- **Ubicacion:** Es la dirección del hospital.
- **CamasDisp:** Indica el número de camas se encuentran disponibles en el hospital.
- **EspDisp:** Es el número de lugares disponibles (en zona de consulta no en el área de hospitalización).

Hospitales_4 contiene los siguientes campos:

- **Clv:** Es un elemento identificador único para cada hospital.
- **NomHosp:** Es el nombre del hospital.
- **Ubicacion:** Es la dirección del hospital.
- **Presupuesto:** Es el dinero que recibe el hospital para cubrir gastos operativos.
- **AñoFundacion:** Es el año en el que se fundó el hospital.
- **numTrabajadores:** Es el número de personas que laboran en el hospital.

Inicialmente el administrador debe definir un esquema mediado y sus respectivos mapeos, por fines prácticos considere que el administrador del sistema ha definido el esquema mediado que se exhibe en la siguiente figura (Véase Figura 6.8) para cubrir las necesidades de información del sistema (se buscó que se asemejara al esquema contemplado en el Apéndice F). Mediante este esquema todos los usuarios pueden visualizar una representación simplificada de los datos que resultan provechosos para sus intereses, sin embargo, esta transparencia delega al sistema integrador la responsabilidad de interactuar con las fuentes remotas, así como todo lo que esto implica.

El trabajo desarrollado promete ser de utilidad dadas las circunstancias descritas en este escenario, porque es necesario interactuar con fuentes de datos que proveen de un acceso limitado y e tienen que formular consultas en términos de las fuentes de datos remotas, con el fin de obtener información de suma importancia por lo que resulta conveniente obtener estos datos lo más rápido posible.

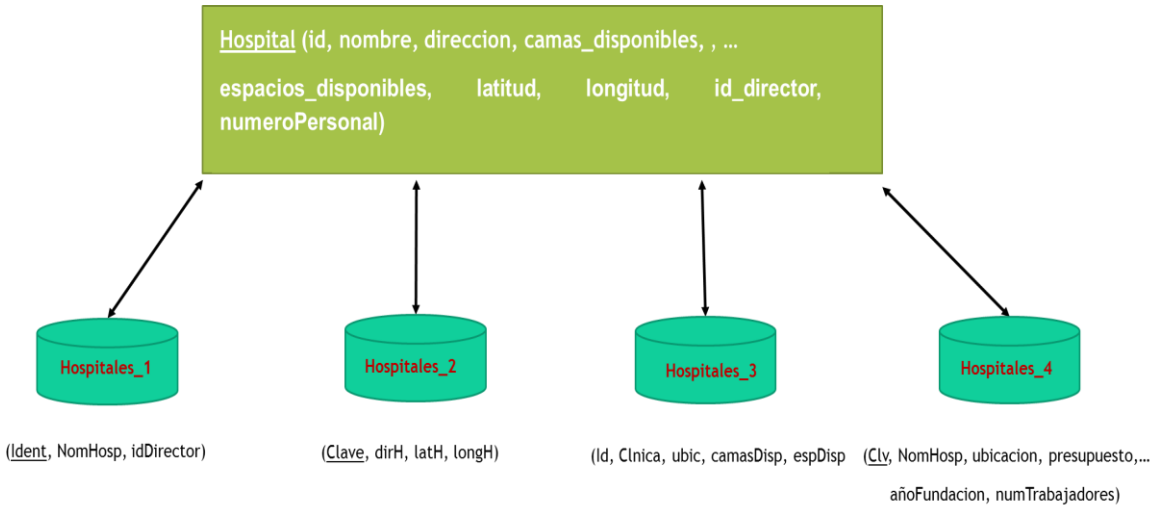


Figura 6.8 Esquema mediado propuesto.

Los mapeos del esquema mediado quedan representados de la siguiente manera (Véase Tabla 6.16).

Tabla 6.16 Mapeos del esquema mediado propuesto.

Campos del esquema mediado	Campos de las bases de datos operativas
id_director	Hospitales_1.id_director
dirección	Hospitales_2.dirH
latitud	Hospitales_2.latH
longitud	Hospitales_2.longH
camasDisp	Hospitales_3. camasDisp
espDisp	Hospitales_3 espDisp
nombre	Hospitales_4.nomHosp
numPersonal	Hospitales_4.numTrabajadores
Id	Hospitales_1.ident Hospitales_2.clave Hospitales_3.id Hospitales_4.clv

A continuación se presentan dos consultas de ejemplo bajo este escenario, sin embargo, en estas no se explora la optimización porque el reducido número de fuentes de datos permite llevar solo tres reuniones y dado que se cuentan con muy pocos registros, pese a ello se muestran con el fin de ver la utilidad de la aplicación para llevar a cabo la descomposición de la consulta mediada en expresiones de las fuentes de datos remotas.

6.3.1. Consulta 1.

Considerando el esquema mediado previamente mostrado, suponga que desea conocer el nombre de los hospitales y su dirección, de aquellos que dispongan de más de 25 camas disponibles y más de 50 empleados.

Campos a visualizar: Hospital.nombre, Hospital.direccion

Condiciones:

C_1: Hospital.camasDisp > 25

C_2: Hospital.numPersonal > 50

Al utilizarse el algoritmo de descomposición de la consulta se obtienen las siguientes consultas objetivo:

- **Q_1:** USE H3@ SELECT Hospitales_3.id FROM H3.Hospitales_3 WHERE H3.Hospitales_3.camasDisp > 25;
- **Q_2:** USE H4@ SELECT Hospitales_4.clv, Hospitales_4.NomHosp FROM H4.Hospitales_4 WHERE H4. Hospitales_4.numTrabajadores > 50

En este caso al haber solo dos consultas objetivo, el programa únicamente busca alguna correspondencia mediante la cual poder llevar a cabo la reunión de dichos datos (Véase Tabla 6.17).

Tabla 6.17 Correspondencias existentes entre las bases de datos.

Correspondencia	Base 1	Base 2
ID="2"	H3. Hospitales_3.id	H4. Hospitales_4.clv

6.3.2. Consulta 2.

Ahora suponga que se desea conocer la dirección, el número de camas disponibles, el número de espacios libres y el número de personas que trabajan en el hospital Juárez.

Campos a visualizar: Hospital.direccion, Hospital.camasDisp, Hospital.espDisp

Condiciones:

C_1: : Hospital.nombre = “Juarez”

Una vez aplicado el algoritmo de descomposición de la consulta se tienen las siguientes consultas:

- **Q_1:** USE H2@ SELECT Hospitales_2.clave, Hospitales_2.DirH FROM H2.Hospitales_2;
- **Q_2:** USE H3@ SELECT Hospitales_3.clv, Hospitales_3.camasDisp, Hospitales_3.espDisp FROM H3.Hospitales_3;
- **Q_3:** USE H4@ SELECT Hospitales_4.clvFROM H4.Hospitales_4 WHERE H4.Hospitales_4.nomHosp = “Juarez”

En la tabla (Véase Tabla 6.18) se muestran las correspondencias existentes para llevar a cabo la combinación de los datos de las consultas objetivo obtenidas y un valor aproximado de las tuplas que dicha operación podría devolver.

Tabla 6.18 Estimación del número de tuplas de los resultados intermedios.

Correspondencia	Base 1	Base 2	Número calculado de tuplas resultantes
ID="1"	H3. Hospitales_3.id	H2.Hospitales_2.clave	6
ID="2"	H3. Hospitales_3.id	H4. Hospitales_4.clv	1
ID="4"	H2.Hospitales_2.clave	H4. Hospitales_4.clv	1

De las anteriores opciones, el prototipo opta por la ejecución de la correspondencia con ID = “4”, debido a que esta implica la ejecución de la consulta objetivo Q_3, en la que se aplica una condición de filtrado de igualdad sobre un campo sin valores repetidos, por lo que la realización de esta condición producirá una sola tupla, posteriormente opta por utilizar la correspondencia ID = “2” para combinar los resultados intermedios con los de la consulta realizada al sitio H3.

6.4 Resultados.

Mediante el análisis de los resultados obtenidos es posible ver que el desempeño de los planes de ejecución generados por el prototipo, tienen un rendimiento similar al provisto por los planes de MySQL, en los dos primeros ejemplos se puede ver que se procesa un número de tuplas un poco menor al de su contraparte y en el tercer ejemplo se puede ver que el número de tuplas procesadas en cada operación de reunión es el mismo. En ninguno de estos planes el plan propuesto procesó más tuplas que en plan generado por MySQL.

Resulta difícil definir un tipo de consulta, en la que la aproximación propuesta siempre obtendrá los mejores resultados, dado que existen varios factores, por ejemplo el tipo de condición de filtrado utilizada, la distribución de los datos, los tipos de datos involucrados en estas, entre otros.

El enfoque heurístico de optimización afirma que se deben de realizar lo más pronto las operaciones de selección y proyección, con el fin de prescindir lo más rápido posible de datos que no contribuyen a solucionar la necesidad de información del usuario. Adaptando este concepto al ambiente operativo de un sistema mediador (en el que este es el único ente capaz de procesar datos de fuentes de datos distintas), resulta conveniente que la cantidad de datos obtenidos de las fuentes remotas sea lo menor posible, sin importar que esto signifique delegar procesamiento a las fuentes de datos remotas, por ello podría afirmarse que las consultas que hacen uso de una o varias condiciones que empleen el operador de igualdad conllevaría a procesar una cantidad menor de tuplas, pero podría ocurrir que la distribución de los valores del dominio del campo sobre el que se especifica una condición de filtrado no sea normal.

Bajo este planteamiento se podría decir que en la mayoría de las veces, una consulta con un conjunto de condiciones de filtrado conjuntivos con operador de igualdad sería el mejor tipo de consulta para la aproximación propuesta.

Capítulo 7 Conclusiones y trabajos futuros.

7.1. Conclusiones.

En este trabajo se buscó desarrollar una alternativa para la generación de planes de ejecución en un sistema mediador (sistema de integración virtual de datos) partiendo de los enfoques utilizados por los sistemas de administración de bases de datos (heurísticas, costos y reglas).

Inicialmente se planteó tomar en consideración los factores referentes a la transmisión de datos en ambientes distribuidos, debido a que el sistema mediador depende completamente de los accesos remotos que realiza a las fuentes de datos operativas (sitios que contienen físicamente los datos a los que accede el sistema mediador), sin embargo por la autonomía de dichos sitios, en estos no se tiene la capacidad de procesar datos provenientes de otras fuentes, por lo que no es posible distribuir la carga de trabajo y recae en el sistema mediador la responsabilidad de realizar las operaciones que involucran datos provenientes de múltiples fuentes.

En la aproximación desarrollada no se consideró explícitamente el tiempo de transmisión de los datos hacia el mediador, porque el tiempo de transmisión es proporcional al tamaño de la tabla, es decir, si el tiempo que tarda en realizarse una reunión es t, entonces el tiempo que tomará realizar la reunión y transmitir sus resultados hasta su punto de destino es kt, donde k es una constante mayor a uno y al realizar la comparación de tiempos solo por la relación menor, multiplicar los tiempos por una constante no altera los resultados de las comparaciones, claro esta afirmación supone que todas las bases de datos tienen la misma constante k, lo que no suele ser cierto, no obstante como se menciona en [22] , esta suele ser una suposición común por los optimizadores en ambientes distribuidos.

Supóngase que se calendarizo mal la reunión de los datos que se tienen con los datos de una base de datos en un punto distante, y esto demora 6 veces más de lo estimado (por no haber considerado el tiempo de transmisión). Se tarda 6t en vez de t, Sin embargo, como de todas maneras se tienen que esperar a que lleguen los datos, no importa el momento en que se solicitó estos datos, el tiempo 6t se suma de cualquier modo al tiempo total de ejecución, sin importar la hora en que se piden. Entonces, tomar muy exactamente los tiempos de transmisión no es muy importante.

Se tomó como enfoque para la generación de planes de ejecución para sistemas mediadores, la reestructuración de la consulta mediada, la generación de consultas bajo los esquemas locales, la transmisión de información por la red de comunicaciones y la combinación de datos. Se optó por desarrollar un método ágil mediante el cual fuera posible reestructurar una consulta mediada en un conjunto de consultas objetivo y el establecimiento de un orden de ejecución mediante la estimación de resultados intermedios.

Para lograr esto se trabajó en aspectos como lo son: la creación del esquema mediado, la localización de correspondencias entre los esquemas locales, la reestructuración de la consulta así como el desarrollo de estimaciones de resultados intermedios.

La generación de planes de ejecución eficientes o cercanos al óptimo resulta ser una tarea difícil, debido a que existe una gran cantidad de variables que pueden afectar la eficiencia de estos, la mayor dificultad es conocer de manera detallada composición y distribución de los datos.

Debido a que, como lo demostraron las pruebas, las estimaciones pueden diferir enormemente a los valores reales; esto podría facilitarse mediante la creación de catálogos más extensos, sin embargo esto podría resultar complicado debido a que las estadísticas especializadas difieren entre un manejador y otro.

Cabe aclarar que a pesar de las limitaciones de las estimaciones con respecto a diferir del valor real, estas proveen de algunos indicios que promueven el desarrollo de planes más eficientes como lo ha indicado hasta ahora los planes de ejecución desarrollados en la etapa de pruebas.

7.2. Trabajos futuros.

Como trabajo futuro se propone la adaptación de la aproximación desarrollada, para que esta sea capaz de procesar otros modelos de datos, aparte del modelo relacional, dado que a pesar de que en la actualidad este modelo predomina en las organizaciones, en años recientes los modelos semiestructurados han obtenido una mayor popularidad.

Además se motiva a que se exploren nuevas alternativas para la estimación de resultados intermedios mediante la consideración del contenido.

Para el algoritmo de *matching* resultaría útil realizar la incorporación de un análisis de contenido así como complementarlo con un diccionario de sinónimos.

7.3. Estancia de investigación.

En el transcurso de mi formación en el CIC-IPN, realice una estancia de investigación en el laboratorio de sistemas distribuidos de la Universidad de Hiroshima en Japón, bajo la dirección del Dr. Satoshi Fujita. Esta tuvo como objetivo aprovechar su dominio en los temas referentes a los sistemas distribuidos para enriquecer el trabajo de investigación mediante la consideración de los factores propios de la red de comunicaciones para la generación de planes de ejecución, sin embargo, pese a obtener conocimientos valiosos, no se logró incorporarlos en el trabajo realizado.

Apéndice A: Archivo del catálogo de las fuentes de datos.

A continuación se muestra como ejemplo el catálogo creado de una fuente de datos llamada "IMDB":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<CONECTION_CATALOGUE DRIVER="com.mysql.jdbc.Driver" NAME="imdb"
PASSWORD="root" URL="jdbc:mysql://localhost:3306/" USER="root">
    <TABLES_CONTAINED DATABASE_NAME="imdb">
        <TABLE_NAME>movies</TABLE_NAME>
        </TABLES_CONTAINED>
        <ATTRIBUTES_TABLE TABLE_NAME="movies" TABLE_SIZE="278">
        <COLUMN_DESCRIPTION COLUMN_NAME="movieName">
        <CARDINALITY>278.0</CARDINALITY>
        <SELECTIVITY>1.0</SELECTIVITY>
        <MAXIMUM_VALUE>Yôjinbô</MAXIMUM_VALUE>
        <MINIMUM_VALUE>12 Angry Men</MINIMUM_VALUE>
        <DATA_TYPE>12</DATA_TYPE>
        <TYPE_NAME>VARCHAR</TYPE_NAME>
        <COLUMN_SIZE>40</COLUMN_SIZE>
        <SQL_DATA_TYPE>0</SQL_DATA_TYPE>
        <IS_NULLABLE>NO</IS_NULLABLE>
        <IS_AUTOINCREMENT>NO</IS_AUTOINCREMENT>
        </COLUMN_DESCRIPTION>
        <COLUMN_DESCRIPTION COLUMN_NAME="protagonistName">
        <CARDINALITY>195.0</CARDINALITY>
        <SELECTIVITY>0.7014</SELECTIVITY>
        <MAXIMUM_VALUE>Yôji Matsuda</MAXIMUM_VALUE>
        <MINIMUM_VALUE>  Marilyn Monroe</MINIMUM_VALUE>
        <DATA_TYPE>12</DATA_TYPE>
        <TYPE_NAME>VARCHAR</TYPE_NAME>
        <COLUMN_SIZE>30</COLUMN_SIZE>
        <SQL_DATA_TYPE>0</SQL_DATA_TYPE>
        <IS_NULLABLE>NO</IS_NULLABLE>
        <IS_AUTOINCREMENT>NO</IS_AUTOINCREMENT>
        </COLUMN_DESCRIPTION>
        <COLUMN_DESCRIPTION COLUMN_NAME="directorName">
        <CARDINALITY>168.0</CARDINALITY>
        <SELECTIVITY>0.6043</SELECTIVITY>
        <MAXIMUM_VALUE>Zack Snyder</MAXIMUM_VALUE>
        <MINIMUM_VALUE>  Stanley Kubrick</MINIMUM_VALUE>
        <DATA_TYPE>12</DATA_TYPE>
        <TYPE_NAME>VARCHAR</TYPE_NAME>
        <COLUMN_SIZE>30</COLUMN_SIZE>
        <SQL_DATA_TYPE>0</SQL_DATA_TYPE>
        <IS_NULLABLE>NO</IS_NULLABLE>
        <IS_AUTOINCREMENT>NO</IS_AUTOINCREMENT>
        </COLUMN_DESCRIPTION>
        <COLUMN_DESCRIPTION COLUMN_NAME="genre">
        <CARDINALITY>17.0</CARDINALITY>
        <SELECTIVITY>0.0612</SELECTIVITY>
        <MAXIMUM_VALUE>Western</MAXIMUM_VALUE>
        <MINIMUM_VALUE> Adventure</MINIMUM_VALUE>
```

```
        <DATA_TYPE>12</DATA_TYPE>
        <TYPE_NAME>VARCHAR</TYPE_NAME>
        <COLUMN_SIZE>15</COLUMN_SIZE>
        <SQL_DATA_TYPE>0</SQL_DATA_TYPE>
        <IS_NULLABLE>NO</IS_NULLABLE>
        <IS_AUTOINCREMENT>NO</IS_AUTOINCREMENT>
        </COLUMN_DESCRIPTION>
        <PRIMARY_KEY PK_COLUMN_NAME="movieName">
        <KEY_SEQ>1</KEY_SEQ>
        <PK_NAME>PRIMARY</PK_NAME>
        </PRIMARY_KEY>
      </ATTRIBUTES_TABLE>
    <ATTRIBUTES_TABLE TABLE_NAME="cinemark" TABLE_SIZE="560"/>
</CONECTION_CATALOGUE>
```

Estructura del archivo del catálogo de una fuente de datos.

Apéndice B: Archivo Correspondencias.

A continuación se muestra la estructura en la que se almacenan las correspondencias.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ROOT>
   <MATCHINGS>
      <ASSOCIATION ID="0" NAME="place_cinema">
        <ELEMENT>
        <DATABASE>c_x</DATABASE>
        <TABLE>cinemex</TABLE>
        <ATTRIBUTE>place</ATTRIBUTE>
        </ELEMENT>
        <ELEMENT>
        <DATABASE>c_y</DATABASE>
        <TABLE>cinepolis</TABLE>
        <ATTRIBUTE>cinema</ATTRIBUTE>
        </ELEMENT>
        </ASSOCIATION>
        <ASSOCIATION ID="1" NAME="movie_title">
        <ELEMENT>
        <DATABASE>c_x</DATABASE>
        <TABLE>cinemex</TABLE>
        <ATTRIBUTE>movie</ATTRIBUTE>
        </ELEMENT>
        <ELEMENT>
        <DATABASE>c_y</DATABASE>
        <TABLE>cinepolis</TABLE>
        <ATTRIBUTE>title</ATTRIBUTE>
        </ELEMENT>
        </ASSOCIATION>
        <ASSOCIATION ID="2" NAME="startTime_startingTime">
        <ELEMENT>
        <DATABASE>c_x</DATABASE>
        <TABLE>cinemex</TABLE>
        <ATTRIBUTE>startTime</ATTRIBUTE>
        </ELEMENT>
        <ELEMENT>
        <DATABASE>c_y</DATABASE>
        <TABLE>cinepolis</TABLE>
        <ATTRIBUTE>startingTime</ATTRIBUTE>
        </ELEMENT>
        </ASSOCIATION><ASSOCIATION ID="3" NAME="cinema_location">
        <ELEMENT>
        <DATABASE>c_y</DATABASE>
        <TABLE>cinepolis</TABLE>
        <ATTRIBUTE>cinema</ATTRIBUTE>
        </ELEMENT>
        <ELEMENT>
        <DATABASE>c_z</DATABASE>
        <TABLE>cinemark</TABLE>
        <ATTRIBUTE>location</ATTRIBUTE>
        </ELEMENT>
        </ASSOCIATION>
```

```
        <ASSOCIATION ID="4" NAME="title_movie">
        <ELEMENT>
        <DATABASE>c_y</DATABASE>
        <TABLE>cinepolis</TABLE>
        <ATTRIBUTE>title</ATTRIBUTE>
        </ELEMENT>
        <ELEMENT>
        <DATABASE>c_z</DATABASE>
        <TABLE>cinemark</TABLE>
        <ATTRIBUTE>movie</ATTRIBUTE>
        </ELEMENT>
        </ASSOCIATION>
        <ASSOCIATION ID="5" NAME="startingTime_startTime">
        <ELEMENT>
        <DATABASE>c_y</DATABASE>
        <TABLE>cinepolis</TABLE>
        <ATTRIBUTE>startingTime</ATTRIBUTE>
        </ELEMENT>
        <ELEMENT>
        <DATABASE>c_z</DATABASE>
        <TABLE>cinemark</TABLE>
        <ATTRIBUTE>startTime</ATTRIBUTE>
        </ELEMENT>
        </ASSOCIATION>
        <ASSOCIATION ID="6" NAME="movie_movie">
        <ELEMENT>
        <DATABASE>c_x</DATABASE>
        <TABLE>cinemex</TABLE>
        <ATTRIBUTE>movie</ATTRIBUTE>
        </ELEMENT>
        <ELEMENT>
        <DATABASE>criticsroundup</DATABASE>
        <TABLE>reviews</TABLE>
        <ATTRIBUTE>movie</ATTRIBUTE>
        </ELEMENT>
        </ASSOCIATION>
        <ASSOCIATION ID="7" NAME="movieName_movie">
        <ELEMENT>
        <DATABASE>imdb</DATABASE>
        <TABLE>movies</TABLE>
        <ATTRIBUTE>movieName</ATTRIBUTE>
        </ELEMENT>
        <ELEMENT>
        <DATABASE>criticsroundup</DATABASE>
        <TABLE>reviews</TABLE>
        <ATTRIBUTE>movie</ATTRIBUTE>
        </ELEMENT>
      </ASSOCIATION>
  </MATCHINGS>
</ROOT>
```

Estructura del archivo que almacena las correspondencias.

Apéndice C: Archivo del esquema mediado.

La información del esquema mediado generado es almacenada en la siguiente estructura:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ROOT>
  <VIRTUAL_ENTITIES>
    <RELATION NAME="Movies">
      <FIELD ID="0" NAME="director" TYPE="SINGLE">
      <ELEMENT ATTRIBUTE="directorName" DATABASE="imdb" TABLE="movies"/>
      </FIELD>
      <FIELD ID="1" NAME="year" TYPE="SINGLE">
      <ELEMENT ATTRIBUTE="premiereYear" DATABASE="criticsroundup" TABLE="reviews"/>
      </FIELD>
      <FIELD ID="2" NAME="genre" TYPE="SINGLE">
      <ELEMENT ATTRIBUTE="genre" DATABASE="imdb" TABLE="movies"/>
      </FIELD>
      <FIELD ID="3" NAME="title" TYPE="MULTIPLE">
      <ELEMENT ATTRIBUTE="movie" DATABASE="c_x" TABLE="cinemex"/>
      <ELEMENT ATTRIBUTE="title" DATABASE="c_y" TABLE="cinepolis"/>
      <ELEMENT ATTRIBUTE="movie" DATABASE="c_z" TABLE="cinemark"/>
      <ELEMENT ATTRIBUTE="movie" DATABASE="criticsroundup" TABLE="reviews"/>
      <ELEMENT ATTRIBUTE="movieName" DATABASE="imdb" TABLE="movies"/>
      </FIELD>
      </RELATION>
      <RELATION NAME="Actors">
      <FIELD ID="4" NAME="protagonistName" TYPE="SINGLE">
      <ELEMENT ATTRIBUTE="protagonistName" DATABASE="imdb" TABLE="movies"/>
      </FIELD>
      <FIELD ID="5" NAME="title" TYPE="MULTIPLE">
      <ELEMENT ATTRIBUTE="movie" DATABASE="c_x" TABLE="cinemex"/>
      <ELEMENT ATTRIBUTE="title" DATABASE="c_y" TABLE="cinepolis"/>
      <ELEMENT ATTRIBUTE="movie" DATABASE="c_z" TABLE="cinemark"/>
      <ELEMENT ATTRIBUTE="movie" DATABASE="criticsroundup" TABLE="reviews"/>
      <ELEMENT ATTRIBUTE="movieName" DATABASE="imdb" TABLE="movies"/>
      </FIELD>
      </RELATION>
      <RELATION NAME="Plays">
      <FIELD ID="6" NAME="location" TYPE="MULTIPLE">
      <ELEMENT ATTRIBUTE="place" DATABASE="c_x" TABLE="cinemex"/>
      <ELEMENT ATTRIBUTE="cinema" DATABASE="c_y" TABLE="cinepolis"/>
      <ELEMENT ATTRIBUTE="location" DATABASE="c_z" TABLE="cinemark"/>
      </FIELD>
      <FIELD ID="7" NAME="startTime" TYPE="MULTIPLE">
      <ELEMENT ATTRIBUTE="startTime" DATABASE="c_x" TABLE="cinemex"/>
      <ELEMENT ATTRIBUTE="startingTime" DATABASE="c_y" TABLE="cinepolis"/>
      <ELEMENT ATTRIBUTE="startTime" DATABASE="c_z" TABLE="cinemark"/>
      </FIELD>
      <FIELD ID="8" NAME="title" TYPE="MULTIPLE">
      <ELEMENT ATTRIBUTE="movie" DATABASE="c_x" TABLE="cinemex"/>
      <ELEMENT ATTRIBUTE="title" DATABASE="c_y" TABLE="cinepolis"/>
      <ELEMENT ATTRIBUTE="movie" DATABASE="c_z" TABLE="cinemark"/>
      <ELEMENT ATTRIBUTE="movie" DATABASE="criticsroundup" TABLE="reviews"/>
```

```
        <ELEMENT ATTRIBUTE="movieName" DATABASE="imdb" TABLE="movies"/>
        </FIELD>
        </RELATION>
        <RELATION NAME="Review">
        <FIELD ID="9" NAME="rating" TYPE="SINGLE">
        <ELEMENT ATTRIBUTE="rating" DATABASE="criticsroundup" TABLE="reviews"/>
        </FIELD>
        <FIELD ID="10" NAME="description" TYPE="SINGLE">
        <ELEMENT ATTRIBUTE="description" DATABASE="criticsroundup" TABLE="reviews"/>
        </FIELD>
        <FIELD ID="11" NAME="title" TYPE="MULTIPLE">
        <ELEMENT ATTRIBUTE="movie" DATABASE="c_x" TABLE="cinemex"/>
        <ELEMENT ATTRIBUTE="title" DATABASE="c_y" TABLE="cinepolis"/>
        <ELEMENT ATTRIBUTE="movie" DATABASE="c_z" TABLE="cinemark"/>
        <ELEMENT ATTRIBUTE="movie" DATABASE="criticsroundup" TABLE="reviews"/>
        <ELEMENT ATTRIBUTE="movieName" DATABASE="imdb" TABLE="movies"/>
        </FIELD>
      </RELATION>
  </VIRTUAL_ENTITIES>
</ROOT>
```

Estructura del archivo del esquema mediado.

Apéndice D: Archivo de la consulta mediada.

La consulta del usuario, la cual esta expresada en el esquema mediado es expresada de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ROOT>
  <CONSULTA_MEDIADA ID="0">
    <PROYECCIONES><
      CAMPO ELEMENTO="title" ENTIDAD="Movie" ID="3"/>
      <CAMPO ELEMENTO="genre" ENTIDAD="Movie" ID="1"/>
      <CAMPO ELEMENTO="location" ENTIDAD="Plays" ID="6"/>
      <CAMPO ELEMENTO="startTime" ENTIDAD="Plays" ID="7"/>
      </PROYECCIONES>
      <CONSULTAS_LOCALES>
      <CONJUNTIVAS>
      <CONSULTA ENTIDAD="Movie">
      <SELECT>
      <CAMPO ELEMENTO="title" ENTIDAD="Movie" ID="3"/>
      <CAMPO ELEMENTO="genre" ENTIDAD="Movie" ID="1"/>
      </SELECT>
      <WHERE>
      <CLAUSULA ID="0" INFORMACION_CAMPO_A="Movie-genre-1"
      INFORMACION_CAMPO_B="1" OPERADOR="EQ" TIPO_CAMPO_A="CAMPO"
      TIPO_CAMPO_B="CONSTANTE"/>
      <CLAUSULA ID="3" INFORMACION_CAMPO_A="Movie-year-2"
      INFORMACION_CAMPO_B="4" OPERADOR="EQ" TIPO_CAMPO_A="CAMPO"
      TIPO_CAMPO_B="CONSTANTE"/>
      <CLAUSULA ID="4" INFORMACION_CAMPO_A="Movie-title-3"
      INFORMACION_CAMPO_B="Plays-title-8" OPERADOR="EQ"
      TIPO_CAMPO_A="CAMPO" TIPO_CAMPO_B="CAMPO"/>
      <CLAUSULA ID="5" INFORMACION_CAMPO_A="Movie-title-3"
      INFORMACION_CAMPO_B="Actors-title-5" OPERADOR="EQ"
      TIPO_CAMPO_A="CAMPO" TIPO_CAMPO_B="CAMPO"/>
      </WHERE>
      </CONSULTA>
      <CONSULTA ENTIDAD="Actors">
      <SELECT/>
      <WHERE>
      <CLAUSULA ID="1" INFORMACION_CAMPO_A="Actors-actor-4"
      INFORMACION_CAMPO_B="2" OPERADOR="EQ" TIPO_CAMPO_A="CAMPO"
      TIPO_CAMPO_B="CONSTANTE"/>
      </WHERE>
      </CONSULTA>
      <CONSULTA ENTIDAD="Plays">
      <SELECT>
      <CAMPO ELEMENTO="location" ENTIDAD="Plays" ID="6"/>
      <CAMPO ELEMENTO="startTime" ENTIDAD="Plays" ID="7"/>
      </SELECT>
      <WHERE>
      <CLAUSULA ID="2" INFORMACION_CAMPO_A="Plays-startTime-7"
      INFORMACION_CAMPO_B="3" OPERADOR="EQ" TIPO_CAMPO_A="CAMPO"
      TIPO_CAMPO_B="CONSTANTE"/>
```

```
          </WHERE>
        </CONSULTA>
      </CONJUNTIVAS>
    <DISYUNTIVAS/>
  </CONSULTAS_LOCALES>
 </ CONSULTA_MEDIADA >
</ROOT>
```

Estructura del archivo de la consulta mediada.

Apéndice E: Archivo del plan de ejecución.

El plan de ejecución es expresado en una estructura similar a la siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ROOT>
  <PLAN_EJECUCION ID="0">
    <PROYECCIONES>
        <CAMPO ELEMENTO="title" ENTIDAD="Movie" ID="3"/>
        <CAMPO ELEMENTO="genre" ENTIDAD="Movie" ID="1"/>
        <CAMPO ELEMENTO="location" ENTIDAD="Plays" ID="6"/>
        <CAMPO ELEMENTO="startTime" ENTIDAD="Plays" ID="7"/>
        </PROYECCIONES>
        <CONSULTAS_LOCALES>
        <CONJUNTIVAS>
        <CONSULTA ENTIDAD="Movie">
        <CONSULTA_FISICA>
        <FUENTE Bd="m">
        <SELECT SELECTIVIDAD_SELECT="0.0011221428571428571">
        <UBICACION ATTRIBUTE="name" DATABASE="m" SELECTIVIDAD="1.0"
        TABLE="movies"/>
        <UBICACION ATTRIBUTE="genre" DATABASE="m" SELECTIVIDAD="0.1571"
        TABLE="movies"/>
        </SELECT>
        <WHERE>
        <CLAUSULA ID="0" INFORMACION_CAMPO_A="Movie-genre-1"
        INFORMACION_CAMPO_B="1" OPERADOR="EQ" SELECTIVIDAD_SELECT="0.1571"
        SELECTIVIDAD_WHERE="0.1571" TIPO_CAMPO_A="CAMPO"
        TIPO_CAMPO_B="CONSTANTE">
        <UBICACION ATTRIBUTE="genre" DATABASE="m" SELECTIVIDAD="0.1571"
        TABLE="movies"/>
        </CLAUSULA>
        <CLAUSULA ID="3" INFORMACION_CAMPO_A="Movie-year-2"
        INFORMACION_CAMPO_B="4" OPERADOR="EQ" SELECTIVIDAD_SELECT="NaN"
        SELECTIVIDAD_WHERE="NaN" TIPO_CAMPO_A="CAMPO"
        TIPO_CAMPO_B="CONSTANTE"/>
        <CLAUSULA ID="4" INFORMACION_CAMPO_A="Movie-title-3"
        INFORMACION_CAMPO_B="Plays-title-8" OPERADOR="EQ"
        SELECTIVIDAD_SELECT="1.0" SELECTIVIDAD_WHERE="1.0"
        TIPO_CAMPO_A="CAMPO" TIPO_CAMPO_B="CAMPO">
        <UBICACION ATTRIBUTE="name" DATABASE="m" SELECTIVIDAD="1.0"
        TABLE="movies"/>
        </CLAUSULA>
        <CLAUSULA ID="5" INFORMACION_CAMPO_A="Movie-title-3"
        INFORMACION_CAMPO_B="Actors-title-5" OPERADOR="EQ"
        SELECTIVIDAD_SELECT="1.0" SELECTIVIDAD_WHERE="1.0"
        TIPO_CAMPO_A="CAMPO" TIPO_CAMPO_B="CAMPO">
        <UBICACION ATTRIBUTE="name" DATABASE="m" SELECTIVIDAD="1.0"
        TABLE="movies"/>
        </CLAUSULA>
        </WHERE>
        </FUENTE>
        <FUENTE Bd="r">
```

```
<SELECT SELECTIVIDAD_SELECT="1.0">
<UBICACION ATTRIBUTE="title" DATABASE="r" SELECTIVIDAD="1.0"
TABLE="review"/>
</SELECT>
<WHERE>
<CLAUSULA ID="0" INFORMACION_CAMPO_A="Movie-genre-1"
INFORMACION_CAMPO_B="1" OPERADOR="EQ" SELECTIVIDAD_SELECT="NaN"
SELECTIVIDAD_WHERE="NaN" TIPO_CAMPO_A="CAMPO"
TIPO_CAMPO_B="CONSTANTE"/>
<CLAUSULA ID="3" INFORMACION_CAMPO_A="Movie-year-2"
INFORMACION_CAMPO_B="4" OPERADOR="EQ" SELECTIVIDAD_SELECT="0.4714"
SELECTIVIDAD_WHERE="0.4714" TIPO_CAMPO_A="CAMPO"
TIPO_CAMPO_B="CONSTANTE">
<UBICACION ATTRIBUTE="premiere" DATABASE="r" SELECTIVIDAD="0.4714"
TABLE="review"/>
</CLAUSULA>
<CLAUSULA ID="4" INFORMACION_CAMPO_A="Movie-title-3"
INFORMACION_CAMPO_B="Plays-title-8" OPERADOR="EQ"
SELECTIVIDAD_SELECT="1.0" SELECTIVIDAD_WHERE="1.0"
TIPO_CAMPO_A="CAMPO" TIPO_CAMPO_B="CAMPO">
<UBICACION ATTRIBUTE="title" DATABASE="r" SELECTIVIDAD="1.0"
TABLE="review"/>
</CLAUSULA>
<CLAUSULA ID="5" INFORMACION_CAMPO_A="Movie-title-3"
INFORMACION_CAMPO_B="Actors-title-5" OPERADOR="EQ"
SELECTIVIDAD_SELECT="1.0" SELECTIVIDAD_WHERE="1.0"
TIPO_CAMPO_A="CAMPO" TIPO_CAMPO_B="CAMPO">
<UBICACION ATTRIBUTE="title" DATABASE="r" SELECTIVIDAD="1.0"
TABLE="review"/>
</CLAUSULA>
</WHERE>
</FUENTE>
</CONSULTA_FISICA>
</CONSULTA>
<CONSULTA ENTIDAD="Actors">
<CONSULTA_FISICA>
<FUENTE Bd="m">
<SELECT SELECTIVIDAD_SELECT="1.0">
<UBICACION ATTRIBUTE="name" DATABASE="m" SELECTIVIDAD="1.0"
TABLE="movies"/>
</SELECT>
<WHERE>
<CLAUSULA ID="1" INFORMACION_CAMPO_A="Actors-actor-4"
INFORMACION_CAMPO_B="2" OPERADOR="EQ" SELECTIVIDAD_SELECT="0.6857"
SELECTIVIDAD_WHERE="0.6857" TIPO_CAMPO_A="CAMPO"
TIPO_CAMPO_B="CONSTANTE">
<UBICACION ATTRIBUTE="actor" DATABASE="m" SELECTIVIDAD="0.6857"
TABLE="movies"/>
</CLAUSULA>
</WHERE>
</FUENTE>
</CONSULTA_FISICA>
</CONSULTA>
<CONSULTA ENTIDAD="Plays">
```

```
<CONSULTA_FISICA>
<FUENTE Bd="c_x">
<SELECT SELECTIVIDAD_SELECT="3.0000000000000003E-4">
<UBICACION ATTRIBUTE="place" DATABASE="c_x" SELECTIVIDAD="0.3"
TABLE="cinemas_x"/>
<UBICACION ATTRIBUTE="startTime" DATABASE="c_x" SELECTIVIDAD="0.9"
TABLE="cinemas_x"/>
<UBICACION ATTRIBUTE="movie" DATABASE="c_x" SELECTIVIDAD="1.0"
TABLE="cinemas_x"/>
</SELECT>
<WHERE>
<CLAUSULA ID="2" INFORMACION_CAMPO_A="Plays-startTime-7"
INFORMACION_CAMPO_B="3" OPERADOR="EQ" SELECTIVIDAD_SELECT="0.9"
SELECTIVIDAD_WHERE="0.9" TIPO_CAMPO_A="CAMPO"
TIPO_CAMPO_B="CONSTANTE">
<UBICACION ATTRIBUTE="startTime" DATABASE="c_x" SELECTIVIDAD="0.9"
TABLE="cinemas_x"/>
</CLAUSULA>
</WHERE>
</FUENTE>
<FUENTE Bd="c_y">
<SELECT SELECTIVIDAD_SELECT="2.5285996055226823E-4">
<UBICACION ATTRIBUTE="cinema" DATABASE="c_y" SELECTIVIDAD="0.3846"
TABLE="cinemas_y"/>
<UBICACION ATTRIBUTE="startingTime" DATABASE="c_y" SELECTIVIDAD="1.0"
TABLE="cinemas_y"/>
<UBICACION ATTRIBUTE="title" DATABASE="c_y" SELECTIVIDAD="1.0"
TABLE="cinemas_y"/>
</SELECT>
<WHERE>
<CLAUSULA ID="2" INFORMACION_CAMPO_A="Plays-startTime-7"
INFORMACION_CAMPO_B="3" OPERADOR="EQ" SELECTIVIDAD_SELECT="1.0"
SELECTIVIDAD_WHERE="1.0" TIPO_CAMPO_A="CAMPO"
TIPO_CAMPO_B="CONSTANTE">
<UBICACION ATTRIBUTE="startingTime" DATABASE="c_y" SELECTIVIDAD="1.0"
TABLE="cinemas_y"/>
</CLAUSULA>
</WHERE>
</FUENTE>
<FUENTE Bd="c_z">
<SELECT SELECTIVIDAD_SELECT="4.107638888888888E-5">
<UBICACION ATTRIBUTE="location" DATABASE="c_z" SELECTIVIDAD="0.35"
TABLE="cinemas_z"/>
<UBICACION ATTRIBUTE="startTime" DATABASE="c_z" SELECTIVIDAD="0.65"
TABLE="cinemas_z"/>
<UBICACION ATTRIBUTE="movie" DATABASE="c_z" SELECTIVIDAD="0.65"
TABLE="cinemas_z"/>
</SELECT>
<WHERE>
<CLAUSULA ID="2" INFORMACION_CAMPO_A="Plays-startTime-7"
INFORMACION_CAMPO_B="3" OPERADOR="EQ" SELECTIVIDAD_SELECT="0.65"
SELECTIVIDAD_WHERE="0.65" TIPO_CAMPO_A="CAMPO"
TIPO_CAMPO_B="CONSTANTE">
```

```
          <UBICACION ATTRIBUTE="startTime" DATABASE="c_z" SELECTIVIDAD="0.65"
          TABLE="cinemas_z"/>
          </CLAUSULA>
          </WHERE>
          </FUENTE>
        </CONSULTA_FISICA>
      </CONSULTA>
    </CONJUNTIVAS>
   <DISYUNTIVAS/>
  </CONSULTAS_LOCALES>
 </PLAN_EJECUCION>
</ROOT>
```

Estructura del archivo del plan de ejecución.

Apéndice F: Aplicación del algoritmo de análisis de similitud con metadatos de fuentes de datos de hospitales.

En este apartado se explora la utilidad del algoritmo de análisis de similitud que se desarrolló en este trabajo, para buscar las posibles correspondencias existentes entre un esquema determinado con otros que manejan el mismo tipo de información.

El esquema usado para esta prueba es una tabla que almacena la información referente a hospitales del sistema RieSis; este software coordina los servicios de emergencia, rescate, atención en hospitales, registro de víctimas, y seguridad pública en caso de una contingencia severa en la ciudad de México. En este software se integra una base de datos en la que se detalla la ubicación de grúas, hospitales y albergues, así como los recursos humanos disponibles para el envío de ayuda. Por ello es necesario este disponga de la mayor cantidad de información disponible, con este fin en este apartado se busca realizar un *matching* entre un esquema definido con respecto al de varias fuentes de datos que manejan información relacionada, con el fin de posteriormente establecer un escenario de integración como el que se planteó en el capítulo de pruebas (Hospitales).

Suponga que se tiene el siguiente esquema global:

Hospital (id, nombre, direccion, camas_disponibles, espacios_disponibles, latitud, longitud, id_director)

A continuación se explica en que consiste cada uno de estos campos:

- **id:** Es el valor utilizado para identificar la información de un hospital.
- **nombre:** Es el nombre del hospital, clínica o centro de salud.
- **dirección:** Es el domicilio donde se encuentra el hospital.
- **camas_disponibles:** Indica el número de camas se encuentran disponibles en el hospital.
- **espacios_disponibles:** Es el número de lugares disponibles (en zona de consulta no en el área de hospitalización).
- **latitud:** Es la latitud de la ubicación del hospital.
- **longitud:** Es la longitud de la ubicación del hospital.
- **id_director:** Es la clave de la persona que está a cargo del hospital.

Inicialmente se plantearon alternativas para el nombrado de los campos que se mencionaron previamente, pensando que bajo esas variantes podrían ser almacenados por otros esquemas que también almacenan información de hospitales, para ello se tomaron en cuenta algunos sinónimos, la abreviación de las palabras, el uso del guion para separar palabras (Véase Tabla F.1).

Tabla F.1. Nombres de campos alternativos a los del sistema RieSis.

Id	Nombre del campo	Variantes del nombre del campo
1	Id	Clve Ident Clave Identificador
2	Nombre	Clínica Hospital NomHosp NomClinica Nom_Hosp Nom_Clinica CentroSalud Centro_Salud NombreClinica NombreHospital Nombre_Clinica Nombre_Hospital
3	Dirección	Sede Lugar Ubicación
4	Camas_Disponibles	Camas CamasLibres Camas_Libres CamasVacantes Camas_Vacantes Camas_Desocupadas CamasDesocupadas
5	Espacios_Disponibles	Espacios EspaciosLibres Espacios_Libres EspaciosVacantes Espacios_Vacantes Espacios_Desocupados EspaciosDesocupados
6	Latitud	Lat DistLat ValorLat
7	Longitud	Longi DistLong ValorLong
8	Id_director	IdDecano IdDirector

		Id_Decano Id_Director IdDirigente CveDecano CveDirector Id_Dirigente Cve_Decano CveDirigente Cve_Director Cve_Dirigente IdAdministrador Id_Administrador CveAdministrador Cve_Administrador

Tomando en cuenta esta información se crearon esquemas que incluyeran estos nombres, posteriormente se utilizó el prototipo para crear los catálogos de dichos esquemas y posteriormente este ejecutó el algoritmo de análisis de similitud usando distintos valores de k, buscando analizar la variación del valor de similitud con respecto a este factor.

A continuación se muestran los resultados obtenidos con K =2, K = 3, K = 4, K = 5 y K = 6 (Véase Tabla F.2, Tabla F.3, Tabla F.4, Tabla F.5, Tabla F.6).

Tabla F.2. Top 35 Correspondencias con mayor grado de similitud con k = 2.

Id	Campo 1	Campo 2	Similitud (%)
1	NomClinica	nombre	16.66
2	IdDirigente	id_director	17.64
3	ValorLat	latitud	18.18
4	CamasDesocupadas	camas_disponibles	19.23
5	Nom_Hosp	nombre	20
6	DistLat	latitud	20
7	CamasVacantes	camas_disponibles	21.73
8	NomHosp	nombre	22.22
9	Ident	id	25
10	Camas	camas_disponibles	25
11	ValorLong	longitud	25
12	Camas_Vacantes	camas_disponibles	26.08
13	DistLong	longitud	27.27
14	Camas_Desocupadas	camas_disponibles	28
15	Id_Decano	id_director	28.57
16	CamasLibres	camas_disponibles	30
17	direccion	IdDirector	30.76
18	Id_Dirigente	id_director	31.25

19	EspaciosDesocupados	espacios_disponibles	32.14
20	Lat	latitud	33.33
21	Camas_Libres	camas_disponibles	35
22	Nombre_Hospital	nombre	35.71
23	EspaciosVacantes	espacios_disponibles	36
24	NombreHospital	nombre	38.46
25	Nombre_Clinica	nombre	38.46
26	Espacios_Vacantes	espacios_disponibles	40
27	Espacios_Desocupados	espacios_disponibles	40.74
28	NombreClinica	nombre	41.66
29	EspaciosLibres	espacios_disponibles	45.45
30	Espacios_Libres	espacios_disponibles	50
31	Espacios	espacios_disponibles	52.94
32	CveDirector	id_director	53.84
33	Longi	longitud	57.14
34	Cve_Director	id_director	61.53
35	IdDirector	id_director	72.72

Tabla F.3. Top 35 Correspondencias con mayor grado de similitud con k = 3.

Id	Campo 1	Campo 2	Similitud (%)
1	Nom_Clinica	nombre	8.33
2	NomClinica	nombre	9.09
3	ValorLat	latitud	10
4	Nom_Hosp	nombre	11.11
5	DistLat	latitud	11.11
6	Cve_Dirigente	id_director	11.11
7	CamasDesocupadas	camas_disponibles	11.53
8	NomHosp	nombre	12.5
9	CamasVacantes	camas_disponibles	13.04
10	CamasLibres	camas_disponibles	14.28
11	Id_Decano	id_director	14.28
12	Camas_Vacantes	camas_disponibles	17.39
13	ValorLong	longitud	18.18
14	Camas_Libres	camas_disponibles	19.04
15	Camas	camas_disponibles	20
16	Camas_Desocupadas	camas_disponibles	20
17	Lat	latitud	20
18	DistLong	longitud	20
19	EspaciosDesocupados	espacios_disponibles	20.68

20	EspaciosVacantes	espacios_disponibles	23.07
21	EspaciosLibres	espacios_disponibles	25
22	direccion	IdDirector	25
23	Id_Dirigente	id_director	26.66
24	Espacios_Vacantes	espacios_disponibles	26.92
25	Espacios_Desocupados	espacios_disponibles	28.57
26	Espacios_Libres	espacios_disponibles	29.16
27	Nombre_Hospital	nombre	30.76
28	NombreHospital	nombre	33.33
29	Nombre_Clinica	nombre	33.33
30	Espacios	espacios_disponibles	33.33
31	NombreClinica	nombre	36.36
32	Longi	longitud	50
33	CveDirector	id_director	50
34	IdDirector	id_director	54.54
35	Cve_Director	id_director	58.33

Tabla F.4. Top 35 Correspondencias con mayor grado de similitud con k = 4.

Id	Campo 1	Campo 2	Similitud (%)
1	Cve_Administrador	id_director	0
2	Cve_Decano	id_director	0
3	CveAdministrador	id_director	0
4	CveDirigente	id_director	0
5	CveDecano	id_director	0
6	id	CveDecano	0
7	latitud	CveDecano	0
8	Cve_Dirigente	id_director	5.88
9	Id_Decano	id_director	7.69
10	CamasDesocupadas	camas_disponibles	8
11	CamasVacantes	camas_disponibles	9.09
12	CamasLibres	camas_disponibles	10
13	ValorLong	longitud	10
14	DistLong	longitud	11.11
15	Camas_Vacantes	camas_disponibles	13.63
16	Camas	camas_disponibles	14.28
17	Camas_Libres	camas_disponibles	15
18	Camas_Desocupadas	camas_disponibles	16.66
19	EspaciosDesocupados	espacios_disponibles	17.85
20	direccion	IdDirector	18.18
21	EspaciosVacantes	espacios_disponibles	20

22	Id_Dirigente	id_director	21.42
23	EspaciosLibres	espacios_disponibles	21.73
24	Espacios_Vacantes	espacios_disponibles	24
25	Nombre_Hospital	nombre	25
26	Espacios_Desocupados	espacios_disponibles	25.92
27	Espacios_Libres	espacios_disponibles	26.08
28	NombreHospital	nombre	27.27
29	Nombre_Clinica	nombre	27.27
30	Espacios	espacios_disponibles	29.41
31	NombreClinica	nombre	30
32	Longi	longitud	40
33	CveDirector	id_director	45.45
34	IdDirector	id_director	50
35	Cve_Director	id_director	54.54

Tabla F.5. Top 35 Correspondencias con mayor grado de similitud con k = 5.

Id	Campo 1	Campo 2	Similitud (%)
1	IdAdministrador	id_director	0
2	IdDirigente	id_director	0
3	IdDecano	id_director	0
4	Cve_Administrador	id_director	0
5	Cve_Dirigente	id_director	0
6	Cve_Decano	id_director	0
7	CveAdministrador	id_director	0
8	CveDirigente	id_director	0
9	CveDecano	id_director	0
10	id	CveDecano	0
11	latitud	CveDecano	0
12	CamasDesocupadas	camas_disponibles	4.16
13	CamasVacantes	camas_disponibles	4.76
14	CamasLibres	camas_disponibles	5.26
15	Camas	camas_disponibles	7.69
16	Camas_Vacantes	camas_disponibles	9.52
17	direccion	IdDirector	10
18	Camas_Libres	camas_disponibles	10.52
19	Camas_Desocupadas	camas_disponibles	13.04
20	EspaciosDesocupados	espacios_disponibles	14.81
21	Id_Dirigente	id_director	15.38
22	EspaciosVacantes	espacios_disponibles	16.66

23	Nombre_Hospital	nombre	18.18
24	EspaciosLibres	espacios_disponibles	18.18
25	NombreHospital	nombre	20
26	Nombre_Clinica	nombre	20
27	Espacios_Vacantes	espacios_disponibles	20.83
28	NombreClinica	nombre	22.22
29	Espacios_Libres	espacios_disponibles	22.72
30	Espacios_Desocupados	espacios_disponibles	23.07
31	Espacios	espacios_disponibles	25
32	Longi	longitud	25
33	CveDirector	id_director	40
34	IdDirector	id_director	44.44
35	Cve_Director	id_director	50

Tabla F.6. Top 35 Correspondencias con mayor grado de similitud con k = 6.

Id	Campo 1	Campo 2	Similitud (%)
1	DistLong	id_director	0
2	ValorLong	id_director	0
3	Id_Administrador	id_director	0
4	Id_Decano	id_director	0
5	IdAdministrador	id_director	0
6	IdDirigente	id_director	0
7	IdDecano	id_director	0
8	Cve_Administrador	id_director	0
9	Cve_Dirigente	id_director	0
10	Cve_Decano	id_director	0
11	CveAdministrador	id_director	0
12	CveDirigente	id_director	0
13	CveDecano	id_director	0
14	id	CveDecano	0
15	direccion	CveDecano	0
16	latitud	CveDecano	0
17	longitud	CveDecano	0
18	Camas_Vacantes	camas_disponibles	5
19	Camas_Libres	camas_disponibles	5.55
20	Id_Dirigente	id_director	8.33
21	Camas_Desocupadas	camas_disponibles	9.09
22	Nombre_Hospital	nombre	10
23	NombreHospital	nombre	11.11

24	Nombre_Clinica	nombre	11.11
25	EspaciosDesocupados	espacios_disponibles	11.53
26	NombreClinica	nombre	12.5
27	EspaciosVacantes	espacios_disponibles	13.04
28	EspaciosLibres	espacios_disponibles	14.28
29	Espacios_Vacantes	espacios_disponibles	17.39
30	Espacios_Libres	espacios_disponibles	19.04
31	Espacios	espacios_disponibles	20
32	Espacios_Desocupados	espacios_disponibles	20
33	CveDirector	id_director	33.33
34	IdDirector	id_director	37.5
35	Cve_Director	id_director	44.44

De los resultados de las pruebas realizadas puede comprobarse que la debilidad de esta aproximación reside en no realizar un análisis del aspecto semántico o de contenido, por ello correspondencias entre campos con nombres como “hospital”, “clínica” o “centroSalud” no son encontradas por esta aproximación.

Sin embargo esta demuestra ser de utilidad en escenarios donde se tienen que analizar elementos con nombres bastante parecidos. Para explorar niveles diferentes de flexibilidad es posible usar distintos valores de k en el proceso de evaluación de posibles correspondencias, entre más cercano sea a uno, mayor será el número de fragmentos analizados lo que conlleva un mayor trabajo en el procesamiento, sin embargo, la evaluación tiende a ser más permisiva, dado que el nivel de análisis se acerca casi al nivel de letra, es decir prácticamente se omitiría analizar la secuencia de las letras.

A continuación, se muestran los niveles de similitud obtenido del algoritmo implementado, al realizar este análisis con los campos cuyo nombre son “Cve_Director” e “Id_Director” (Véase Tabla F.7).

Tabla F.7. Estructura del archivo que almacena las correspondencias.

Campos	k=2	k=3	k=4	k=5	k=6
Cve_Director	61.53%	58.33%	54.54%	50%	44.44%
Id_Director					

En la siguiente tabla se muestra en detalle los factores usados para calcular el valor de similitud del ejemplo anterior (Véase Figura F.8).

Tabla F.8. Detalles del proceso del valor de similitud obtenido.

Valor de K	Intersección	Unión	Similitud
2	[_D, DI, IR, RE, EC, CT, TO, OR]	[CV, VE, E_, _D, DI, IR, RE, EC, CT, TO, OR, ID, D_]	8/13 *100
3	[_DI, DIR, IRE, REC, ECT, CTO, TOR]	[CVE, VE_, E_D, _DI, DIR, IRE, REC, ECT, CTO, TOR, ID_, D_D]	7/12 *100
4	[_DIR, DIRE, IREC, RECT, ECTO, CTOR]	[CVE_, VE_D, E_DI, _DIR, DIRE, IREC, RECT, ECTO, CTOR, ID_D, D_DI]	6/11 *100
5	[_DIRE, DIREC, IRECT, RECTO, ECTOR]	[CVE_D, VE_DI, E_DIR, _DIRE, DIREC, IRECT, RECTO, ECTOR, ID_DI, D_DIR]	5/10 *100
6	[_DIREC, DIRECT, IRECTO, RECTOR]	[CVE_DI, VE_DIR, E_DIRE, _DIREC, DIRECT, IRECTO, RECTOR, ID_DIR, D_DIRE]	4/9 *100

De este ejemplo se ve que el nivel de similitud obtenido por las mismas palabras decrece con respecto al aumento de k, esto se debe a que entre mayor sea k, conlleva a un menor número de fragmentos lo que dificulta obtener un alto puntaje.

Mediante estas pruebas realizadas es posible ver los resultados del análisis de similitud usando el coeficiente de Jaccard con distintos metadatos que suelen ser utilizados para almacenar la misma información, lo que sirve como precedente para futuros empleos de esta técnica para la generación de *mathcings* o correspondencias.

Bibliografía

[1] O. M. Duschka y M. R. Genesereth, «Infomaster - An Information Integration Tool" *In Proceedings of the International Workshop Intelligent Information Integration during the 21st German Annual Conference on Artificial Intelligence,* pp. 9-12, Septiembre 1997.

[2] R. J. J. Bayardo, J. R. J. a. B. W. a. B. R. a. C. A. a. F. J. a. H. A. a. K. V. a. K. T. a. M. G. a. N. M. a. R. M. a. R. M. a. S. R. a. U. C. a. U. A. a. W. D. Bayardo, W. G. Bohrer, R. S. Brice, A. Cichocki, J. Fowler, A. S. Helal, V. Y. Kashyap, T. B. Ksiezyk, G. Martin, M. M. Nodine, M. Rashid, M. . E. Rusinkiewicz, R. Shea, C. Unnikrishnan, . A. J. Unruh y D. Woelk, «InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments" *Proceedings of the 1997 ACM SIGMOD international conference on Management of data* , pp. 195-206, 1997.

[3] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman y J. Widom, «The TSIMMIS Project: Integration of Heterogeneous Information Sources" *IPSJ,* 1994.

[4] T. Kirk, A. Y. Halevy, Y. Sagiv y D. Srivastava, «The Information Manifold" *In Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments,* pp. 85-91, 1995.

[5] Jones Kent, «About Critics Round Up" [En línea]. Available: http://criticsroundup.com/about-critics-round-up/.. [Último acceso: 12 Junio 2014].

[6] G. Graefe y D. J. DeWitt, «The EXODUS Optimizer Generator" *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data,* 1987.

[7] G. Graefe y W. J. McKenna, «The Volcano Optimizer Generator: Extensibility and Efficient Search" *Proceedings of IEEE 9th International Conference on Data Engineering.*, pp. 209-218, 1993.

[8] G. Graefe, «The Cascades Framework for Query Optimization" *IEEE Data Eng. Bull.*, 1995.

[9] T. M. Conelly y C. E. Begg, «Distribuited Databases" de *Database Systems: A Practical Approach to Design, Implementation and Management*, 4th ed., Pearson Addison Wesley, 2004.

[10] C. J. Date, «Procesamiento de consultas" de *Introducción a los sistemas de bases de datos*, 7ª ed., Pearson Prentice Hall, 2001.

[11] R. Elmasri y S. B. Navathe, «Query Processing" de *Fundamental of Database Systems*, 6th ed., Pearson Addison Wesley, 2010.

[12] D. Paresh, V. Virparia, D. Sanjay, H. Buch y R. F. Parabia, «Trade and Tricks: Traditional vs. Virtual Data Warehouse" *International Journal of Advanced Engineering & Application,* January 2010.

[13] M. R. Genesereth , A. M. Keller y O. . M. Duschka, *Infomaster: an information integration system,* 1997, pp. 539-542.

[14] D. Woelk, W. Bohrer, N. Jacobs, K. Ong, C. Tomlinson y . C. Unnikrishnan, «Carnot

and InfoSleuth: database technology and the World Wide Web" *n Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pp. 443-444, 1995.

[15] Y. Arens, C.-N. Hsu y C. . A. Knoblock, «Query Processing in the SIMS Information Mediator" *Advanced Planning Technology AAAI*, pp. 61-69, 1996.

[16] M. . J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, J. Thomas, . J. H y E. . L. Wimmers , «Towards Heterogeneous Multimedia Information Systems: The Garlic Approach" *Research Issues in Data Engineering, 1995: Distributed Object Management, Proceedings. RIDE-DOM '95. Fifth International Workshop*, pp. 124 - 131, 1995.

[17] . D. Calvanese, G. De Giacomo y M. Lenzerini, «Description Logics for Information Integration" *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, pp. 41-60, 2002.

[18] M. Lenzerini, «Data Integration: A Theoretical Perspective" *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 233-246, 2002.

[19] D. Kossmann, «The State of the Art in Distributed Query Processing" *ACM Comput. Surv.*, pp. 422- 469, 2000.

[20] A. Silberschatz, H. F. Korth y S. Sudarshan, «Query Processing" de *Database System Concepts*, 6th ed., Mc Graw Hill, 2011.

[21] Y. E. Ioannidis, «Query Optimization" *ACM Comput. Surv.*, 1996.

[22] . M. O. Tamer y P. Valduriez, «Optimization of Distributed Queries" de *Principles of Distributed Database Systems*, Springer, 2011.

[23] A. Swami y A. Gupta, «Optimization of Large Join Queries" pp. 8-17, Junio 1988.

[24] A. N. Swami, «Optimization of large join queries: combining heuristics and combinatorial techniques» *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pp. 367-376, Junio 1989.

[25] Y. E. Ioannidis y E. Wong, «Query optimization by simulated annealing» *SIGMOD '87 Proceedings of the 1987 ACM SIGMOD international conference on Management of data*, pp. 9-22, Mayo 1987.

[26] Y. E. Ioannidis y Y. Kang, «Randomized algorithms for optimizing large join queries» *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pp. 312-321, Mayo 1990.

[27] Y. C. Kang, *Randomized Algorithms for Query Optimization*, Madison, 1991.

[28] Y. H. . y S. Lafortune, «An Intelligent Search Method for Query Optimization by Semijoins» *IEEE Trans. on Knowl. and Data Eng.*, pp. 226--237, 1989.

[29] Needham Col, «IMDB» [En línea]. Available: http://www.imdb.com/help/show_leaf?history.. [Último acceso: 12 Junio 2014].

[30] A. . Y. Halevy, Z. G. Ives y A. Doan, «Query Processing» de *Principles of Data Integration*, 1st ed., Morgan Kaufmann, 2012.