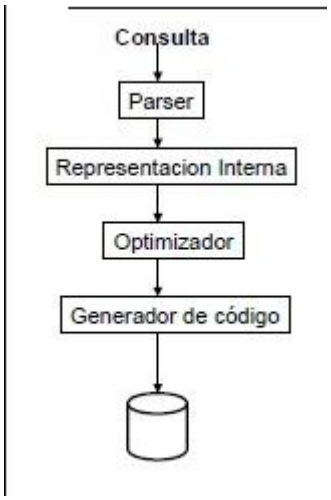
# Unidad III

**Procesamiento de consultas distribuidas.**

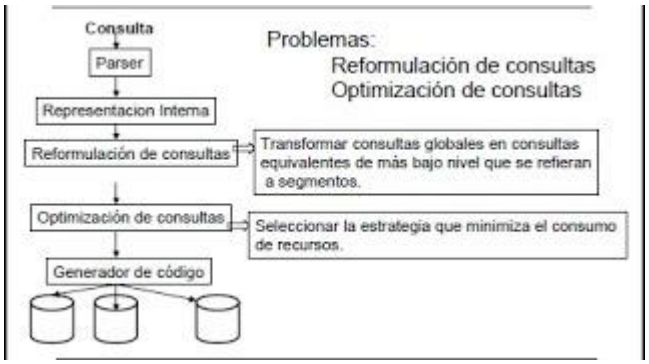
### 3.1 Metodología del procesamiento de consultas distribuidas.

Primeramente se debe de contar con heterogenidad de los datos, para que puedan ser usados para formular consultas. Tenemos los sigueintes ejemplos:

BD CENTRALIZADA




BD DISTRUIBUIDA




Asi como tambien necesitamos contar con:

-Localizacion de los datos para generar reglas heuristicas
-Descomposicion de consultas en paralelo en cada nodo
-Reducir la cantidad de datos a transferir en la red

### 3.2 Estrategias de procesamiento de consultas distribuidas.

Contamos con la estategia de Reformulacion de consultas, que nos sirve para encontrar q la informacion que nos va a proveer sea solo la que se le pidio por la fuente

Tambien se cuenta con la estrategia de descomposicion de las fuentes, q consiste en que segun las fuentes q pidan cierto tipo de datos sean las atentidas con mayor velocidad.

#### 3.2.1 Árboles de consultas.

- *Arbol de consulta que es una estructura de árbol que corresponde a una expresión del álgebra relacional en el que las tablas se representan como nodos hojas y las operaciones del álgebra relacional como nodos intermedios.*

#### 3.2.2 Transformaciones equivalentes.

Cuando una base de datos se encuentra en múltiples servidores y distribuye a un número determinado de nodos tenemos:
1.-el servidor recibe una petición de un nodo
2.-el servidor es atacado por el acceso concurrente a la base de datos cargada localmente
3.-el servidor muestra un resultado y le da un hilo a cada una de las maquinas nodo de la red local.
Cuando una base de datos es accesada de esta manera la técnica que se utiliza es la de fragmentación de datos que puede ser hibrida, horizontal y vertical.
En esta fragmentación lo que no se quiere es perder la consistencia de los datos, por lo tanto se respetan las formas normales de la base de datos.
Para realizar una transformación en la consulta primero se desfragmenta siguiendo los estándares marcados por las reglas formales y posteriormente se realiza el envió y la máquina que recibe es la que muestra el resultado pertinente

para el usuario, de esta se puede producir una copia que será la equivalente a la original.

### 3.2.3 Métodos de ejecución del Join.

Sean (R) y s(S) dos relaciones:

Si R S= entonces r s es lo mismo que r x s, y por lo tanto se puede utilizar la estimación del producto cartesiano.

Si R S es una clave de R entonces el número de tuplas en r s no es mayor que el número de tuplas en S. Si R S es una clave externa de R entonces el número de tuplas de r s es exactamente el número de tuplas de S.

Si R S no es clave de R ni de S entonces se supone que cada valor aparece con la misma probabilidad , por lo tanto, sea t una tupla de r y sea R S=Ā, entonces se estima que la tupla t produce :

tuplas en s, por lo tanto se estima el tamaño de r s = (a) al cambiar los papeles de r y s se tiene (b)

Estos valores serán distintos si y sólo si V(A,r) V(A,s), si este es el caso, la más baja estimación de ambas será la más conveniente.

Join en bucles anidados.

Si z = r s, r recibirá el nombre de relación externa y s se llamará relación interna, el algoritmo de bucles anidados se puede presentar como sigue.

para cada tupla tr en rpara cada tupla ts en ssi (tr,ts) satisface la condición entonces añadir tr ts al resultado Algoritmo 5–1 - Join en bucles anidados.

Donde tr ts será la concatenación de las tuplas tr y ts .

Como para cada registro de r se tiene que realizar una exploración completa de s, y suponiendo el peor caso, en el cual la memoria intermedia sólo puede concatenar un bloque de cada relación, entonces el número de bloques a acceder es de . Por otro lado, en el mejor de los casos si se pueden contener ambas relaciones en la memoria intermedia entonces sólo se necesitarían accesos a bloques.

Ahora bien, si la más pequeña de ambas relaciones cabe completamente en la memoria, es conveniente utilizar esta relación como la relación interna, utilizando así sólo accesos a bloques.

Join en bucles anidados por bloques.

Una variante del algoritmo anterior puede lograr un ahorro en el acceso a bloques si se procesan las relaciones por bloques en vez de por tuplas.

para cada bloque Br de rpara cada bloque Bs de spara cada tupla tr en Brpara cada tupla ts en Bssi (tr,ts) satisface la condición entonces añadir tr ts al resultado

Algoritmo 5–2 - Join en bucles anidados por bloques.

La diferencia principal en costos de este algoritmo con el anterior es que en el peor de los casos cada bloque de la relación interna s se lee una vez por cada bloque de r y no por cada tupla de la relación externa, de este modo el número de bloques a acceder es de donde además resulta más conveniente utilizar la relación más pequeña como la relación externa.

Join en bucles anidados por índices.

Este algoritmo simplemente sustituye las búsquedas en tablas por búsquedas en índices, esto puede ocurrir siempre y cuando exista un índice en el atributo de join de la relación interna. Este método se utiliza cuando existen índices así como cuando se crean índices temporales con el único propósito de evaluar la reunión.

El costo de este algoritmo se puede calcular como sigue: para cada tupla de la relación externa r se realiza una búsqueda en el índice de s para recuperar las tuplas apropiadas, sea c = costo de la búsqueda en el índice, el cual se puede calcular con cualquiera de los algoritmos A3, A4 o A5. Entonces el costo del join es ; si hay índices disponibles para el atributo de join en ambas relaciones, es conveniente utilizar la relación con menos tuplas.

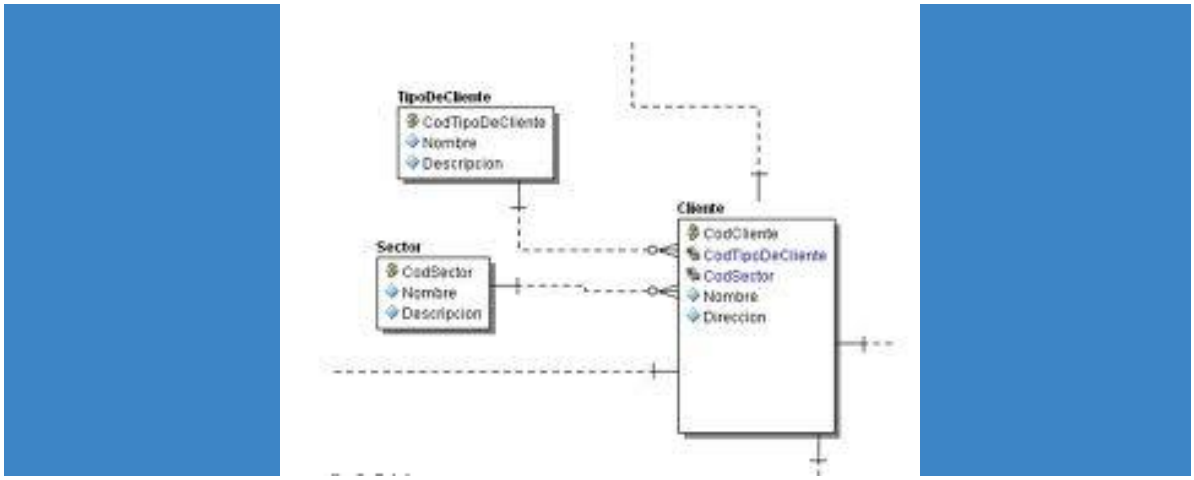
### 3.3 Optimización de consultas.

Para poder optimizar una consulta necesitamos tener claras las propiedades del algebra relacional para asegurar la reformulacion de la consulta, al optimizar una consulta obtenemos los siguientes beneficios:

-minimizar costos

-Reducir espacios de comunicaciones

-Seguridad en envios de informacion




### 3.3.1 Optimización global de consultas.

Cuando hablamos de **optimización de consultas** nos referimos a mejorar los tiempos de respuesta en un sistema de gestión de bases de datos relacional, pues la optimización es el proceso de modificar un sistema para mejorar su eficiencia o también el uso de los recursos disponibles.

En bases de datos relacionales el lenguaje de consultas SQL es el más utilizado por el común de los programadores y desarrolladores para obtener información desde la base de datos. La complejidad que pueden alcanzar algunas consultas puede ser tal, que el diseño de una consulta puede tomar un tiempo considerable, obteniendo no siempre una respuesta óptima.

### 3.3.2 Optimización local de consultas.

Para procesar una consulta local solo se hace referencia a tablas y bases de datos locales(no a vistas ni a tablas remotas) , es decir que estén dentro de la misma instancia delmanejador de bases de datos, en una única maquina y que nos se tenga que conectar alservidor a otras maquinas para poder efectuar la consulta.Cada subconsulta que se ejecuta en un nodo, llamada consulta local, es optimizada usandoel esquema local del nodo. Hasta este momento, se pueden

eligen los algoritmos
pararealizar las operaciones relacionales. La optimización local utiliza los algoritmo
s desistemas centralizados